サーバーベースストレージにおける ワークロードを考慮した容量効率改善手法

深谷　崇元[†,††]　　Le Hieu Hanh[†]　　横田　治夫[†]

†† 株式会社日立製作所　〒 244-0817 神奈川県横浜市戸塚区吉田町 292 番地
† 東京工業大学　〒 152-8552 東京都目黒区大岡山 2-12-2
E-mail: *takayuki.fukatani.re@hitachi.com, hanhlh@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp*

あらまし　安価な汎用サーバをストレージとして使用するサーバーベースストレージが、専用ストレージアプライアンスの代替として広がっている。低信頼な汎用サーバで高信頼なストレージを実現するには、サーバ間のデータ冗長化が必要となり、冗長データの増加に伴う高コスト化が問題となる。従来研究では、データの冗長化方式を、アクセス頻度に応じて更新性能の高いレプリケーションと容量効率のよい Erasure coding(EC）を切り替える冗長化制御が提案されている。しかしながら、従来の固定長ベースの冗長化制御は、細粒度の更新偏りがあるワークロードに対して、容量削減効果が小さくなるという問題があった。本研究では、ワークロードの更新偏りを考慮した容量効率改善手法を提案する。提案方式では、細粒度の更新偏りがあるワークロードに対して、更新差分をデータ本体と分けて管理する差分データ管理を行うことで EC による性能低下を抑制し、より多くのデータの EC 化を可能とした。また、シミュレーションを用いた評価により、提案方式が TPC-C ベンチマークにおける容量削減効果を、従来方式に対して最大 2.17 倍まで改善することを確認した。

キーワード　ストレージ，冗長化，Erasure Coding

1 はじめに

近年、安価な汎用サーバーをソフトウェア制御により高信頼化しストレージとして使用する、サーバーベースストレージが広まっている [1]。サーバーベースストレージは、安価な汎用サーバーを使用することで従来のアプライアンス型の専用ストレージに対し、低コストなシステムを実現する。

障害発生頻度が高い低信頼な汎用サーバーを高信頼化するには、複数サーバーをまたがったデータ冗長化が必要となる [2]。このようなデータ冗長化手法として、アプリケーションや OS によるサーバー間レプリケーション [3] [4] や、Ceph や GlusterFS [5] [6] などの分散ストレージを用いたデータ冗長化が広く使われている。

このようにデータ冗長化を必要とするサーバーベースストレージでは、冗長データの容量消費に伴うシステムコスト増加が問題となる。サーバー間レプリケーションは、ローカルに格納したユーザーデータの複製を、一台以上の別のサーバーに格納するため、冗長度に比例して複製データの容量が増えてしまう。一方、Erasure coding(EC) を用いれば、複製データを小容量のパリティデータに置き換えることで容量効率を改善できる [7]。しかし、EC はパリティ演算やデータの分散格納に伴う性能オーバヘッドを伴い、性能要件が高いケースには適用できない。

従来研究では、更新性能の高いレプリケーションと、容量効率のよい EC を、更新頻度に基づき切り替える動的な冗長化制御が提案されている [5] [8]。このような動的な冗長化制御はデータに対するアクセスを監視し、アクセスが多いデータ（ホットデータ）をレプリケーションで、アクセスが少ないデータ (コールドデータ) を EC で冗長化することで、性能を維持した容量効率改善を実現する。

しかしながら、従来方式では細粒度の更新偏りが発生した場合に、容量削減効果が小さくなるという課題がある。従来方式では、アクセス監視の単位より小さい単位で更新の偏りが発生した場合、細粒度のコールドデータを検出することができない。そのため、コールドデータの割合が大きかったとしても、EC 化できるデータ容量が小さくなり、容量効率も低下する。例えば、オンライントランザクション処理 (OLTP) やデータ解析用の非構造 DB では、細粒度のデータ構造に対し更新が発生するため、容量効率が低下してしまう。

本研究では、サーバーベースストレージにおけるワークロードの更新偏りを考慮した容量効率改善手法を提案する。提案方式では、細粒度の更新偏りがあるデータに対し、更新による差分データを EC 化したデータ本体と分けて管理する差分データ管理を導入する。差分データをレプリケーションにより冗長化することで、EC 化したデータに対する更新性能低下を抑制することができる。また、更新偏りの大きなデータを、差分データ管理ありで EC 化することで、より多くのデータの EC 化が可能となり、容量効率を改善することができる。

また、提案方式の有効性を確認するために、アクセス性能評価と容量削減効果の評価を行った。プロトタイプを用いたアクセス性能評価では、提案方式が従来方式に対して、EC 化したデータへのアクセス性能を最大 236%改善することを確認した。シミュレーションを用いた容量削減効果の評価では、TPC-C

ベンチマークを実行した場合に、従来方式に対して最大 2.17 倍の容量削減を実現することを確認した。

2 従来技術

サーバーベースストレージにおける容量削減技術として、ユーザーデータの冗長化方式をワークロードに応じてレプリケーションと EC 間で切り替える動的冗長化制御が提案されている。動的冗長化制御はその冗長化制御のデータ制御単位により、ファイル・オブジェクト単位の冗長化制御と、より細粒度な固定長チャンク単位の冗長化制御に分類できる。本章では、これら従来技術の概要と課題について述べる。

2.1 動的冗長化制御方式

2.1.1 ファイル・オブジェクト単位冗長化制御

Ceph [5] や GlusterFS [6] などの分散ストレージでは、異なるアクセス特性を持ったストレージを組み合わせ、ワークロードに応じてデータの格納先を切り替える階層ストレージ機能を提供する。上位階層のデータの冗長化方式をレプリケーション、下位階層の冗長化方式を EC とする階層ストレージを構成した場合、ワークロードに応じてユーザーデータの冗長化方式をレプリケーション、EC 間で切り替えることが可能となる。その結果、アクセス頻度の高いユーザーデータの冗長化方式を高性能なレプリケーション、それ以外のデータの冗長化方式を EC とすることで、性能を維持した容量効率を実現できる。

これらの手法では、ファイル・オブジェクト単位でデータへのアクセス数やアクセス時刻を記録し、階層移動対象データの選択に使用する。そのため、ファイル・オブジェクト内でホットデータとコールドデータが混在した場合に、ファイルやオブジェクト内の一部のデータのみの冗長化方式を変更することはできない。その結果、同レベルのアクセス性能を実現するために必要なレプリケーションデータの容量が増え、容量効率が低くなる点が問題となる。

2.1.2 チャンク単位冗長化制御

Dynamic Redundancy Control(DRC) [8], [9] などのストレージ制御では、ユーザーデータを固定長のデータ単位であるチャンクにわけ、チャンク単位でデータの冗長化方式を制御する。これらのストレージ制御では、チャンクのアクセス頻度に応じて冗長化方式をレプリケーションと EC で切り替える。ファイル内にホットデータとコールドデータが混在した場合でも、チャンクサイズ単位で冗長化方式を切り替えられるため、ファイル・オブジェクト単位の冗長化制御に対して高い容量効率を実現できる。

一方でチャンク単位の冗長化制御を用いた場合、モニタデータによるメモリ消費やメタデータによる容量消費を考慮する必要がある。チャンク単位の冗長化制御では、チャンクに対するアクセスをモニタし、EC 化するチャンクの選択に使用する。これらのモニタデータや冗長化のためのメタデータは、チャンクごとに必要となり、チャンク数に比例したメモリ容量消費とディスク容量消費を伴う。より小さなチャンクサイズを用いた場合、より細粒度の冗長化制御が可能となるが、チャンク数が増え、メモリ消費や容量消費が大きくなる点が問題となる。

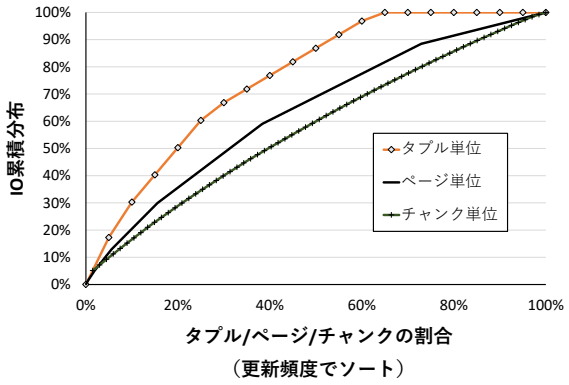

図 1 PostgreSQL の更新 IO 分布

2.2 従来技術の課題

従来技術では、冗長化制御の単位となる冗長化単位ごとにアクセス頻度をモニタし、モニタデータを EC 化するデータの選択に使用する。そのため、冗長化単位より小さい粒度で更新偏りが発生した場合に、ホットデータと近接するコールドデータを EC 化できず容量効率が悪くなる点が課題となる。これは、ファイル・オブジェクト、またはチャンクなどの冗長化単位の一部にホットデータが含まれていた場合、それらのデータ全体をホットデータとして扱うためである。

一方、先行研究 [10] により、多くのワークロードではアプリケーションの管理するデータ単位で更新偏りが発生することが知られている。これは、アプリケーションによるデータ更新が、そのアプリケーションの管理するデータ単位に発生するためである。これらの更新偏りがあるワークロードでは、一部のデータに大部分のアクセスが集中する特性をもつことが多く、多くのアプリケーションの更新は Zipf 分布に従う。また、実際の OLTP を模擬した TPC-C ベンチマークも、更新偏りを前提とした Non-uniform random 分布 [11] に従ったアクセスパターンを使用している。

動的冗長化制御の容量効率に対する更新偏りの影響を調べるため、TPC-C ベンチマークを用いた事前実験を実施した。事前実験では、PostgreSQL [12] に対して TPC-C ベンチマークを実行した際の IO トレースを採取し、PostgreSQL の管理データ単位と、DRC が用いる 512KB チャンクあたりの更新アクセス分布を調べた。事前実験の結果を図 1 に示す。

グラフは TPC-C ワークロードで使用するテーブルのうち、更新アクセスが多い stock テーブルにおける、PostgreSQL の管理データ単位である 8KB ページと 512KB チャンクの IO 累積分布を示している。累積分布は更新アクセス頻度に従ってページ、チャンクをソートした場合の結果を示しており、更新アクセスの割合に対して、実際にアクセスが発生するページ、チャンクの割合を確認することができる。また、Non-uniform

random 分布に基づいて計算した、タプルの IO 累積分布を合わせて示している。

グラフからわかるようにタプル単位では、90%の更新アクセスが上位 5 割のタプルに集中しているのに対し、チャンク単位では上位 9 割近くのチャンクにわたって発生している。この結果は、更新アクセス比率が 10%以下のデータをコールドデータと判定した場合に、タプル単位では約 5 割のデータを EC 化できるのに対し、チャンク単位では 1 割以下のデータしか EC 化できないことを示している。実際には半分のデータがコールドデータがあるにも関わらず、従来のチャンク単位の冗長化制御では 1 割以下のデータしか EC 化できず、容量削減量が小さくなっていることを意味する。

これらの結果から、従来の固定長チャンクを用いた動的冗長化制御では、細粒度の更新偏りが発生した場合に、EC 化できるデータ容量が小さくなり、容量効率改善効果が小さくなることがわかる。このようなワークロードの例として、OLTP やデータ解析用の非構造 DB があげられる。これらのアプリケーションでは、小サイズの管理データ構造に対し更新が発生するため、細粒度の更新偏りが発生する。高価な Nonvolatile memory express solid-state drives (NVMe SSDs) や Storage class memory(SCM) を多用する OLTP や、大容量非構造データを格納するデータ解析での容量効率低下は、システムコストの大幅増加につながる。

3 目的とアプローチ

本研究はサーバーベースストレージにおいて、細粒度の更新偏りがあるワークロードに対して、性能を維持した容量効率改善を実現することを目的とする。本研究では目的達成にあたり、以下の 2 つのアプローチによる従来方式の改善を行う。

a) EC 化したチャンクの差分データ管理

EC 化したデータに対して更新アクセスが多発した場合、パリティ更新のオーバヘッドにより大幅な性能低下が発生する。従来方式では更新頻度の低いチャンクを優先して EC 化することで、EC 化に伴う性能低下を回避していた。そのため、チャンクの一部にでも更新頻度の高いデータを含んでいた場合、内部にコールドデータを含んでいたとしても EC 化することができないため、容量効率が低下してしまう。

提案方式では、EC 化したチャンクに対する差分データをデータ本体と別に管理する差分データ管理を導入する。差分データの冗長化方式をレプリケーションとすることで、データ更新時のパリティ更新を不要とし、EC 化による更新性能低下を回避することができる。差分データ管理により、EC 化したチャンクに対するアクセス性能低下を回避できるため、より多くのデータを EC 化できるようになる。

b) 更新偏りを考慮した EC 化制御

差分データ管理を行うことで EC 化による性能低下を回避できるが、差分データによる容量消費が発生する。更新されるデータの割合が大きなチャンクに対して差分データ管理を行った場合、差分データ容量が EC 化により削減できる容量よりも大きくなり、EC 化しても容量効率が逆に悪化することになる。そのため、差分データ管理を行うチャンクは、差分データの容量が小さいチャンクに限定する必要がある。

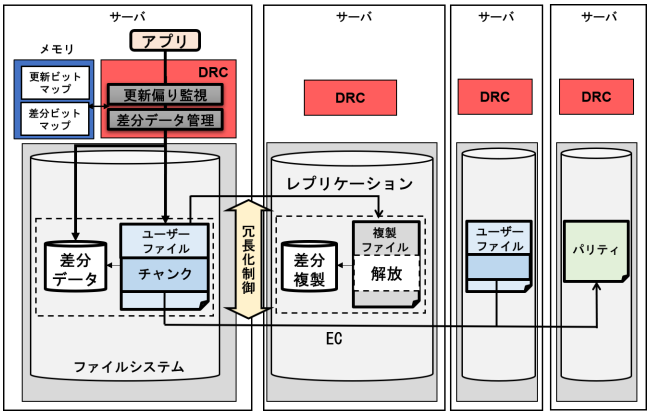

図 2 提案方式概要

提案方式では従来のモニタリングに加え、更新偏りを検出するための細粒度の更新監視を新たに追加する。更新偏りの大きさから、差分データ容量を見積もることで、差分データ管理による容量削減可否の判定が可能となる。更新偏りが大きいチャンクを、差分データ管理付きの EC 化対象とすることで、容量効率を高めることができる。

4 提案方式

本章では従来技術の改善方式として、ワークロードを考慮した容量効率改善手法を提案する。

4.1 方式概要

提案方式は、従来方式に対し、EC 化したチャンクの差分データ管理と、ページ単位の更新監視による更新偏りの検出を新たに導入する。提案方式は、更新偏りの大きなチャンクを、差分データ管理付きで EC 化することで、従来技術に対して更なる容量効率改善を実現する。提案方式の概要を図 2 に示す。

提案方式のベースとなる DRC はローカルファイルシステムの上位で動作する冗長化制御層である。DRC は、ローカルファイルシステムに格納したユーザーファイルに対して、1 台以上のレプリケーション・ペアと呼ぶサーバ上に複製ファイルを格納することでデータを冗長化する。また、低アクセス頻度のチャンクをサーバ間で EC 化し、複製データをパリティデータに置き換えることで、冗長データの容量を削減する。

差分データ管理は、EC 化したチャンクに対する更新データをデータ本体と別領域に複製して格納するデータ管理方式である。DRC は、差分データ管理を行うチャンクに対して更新があった場合に、差分データをローカルファイルシステムの別領域に格納し、レプリケーション・ペアに複製する。EC 化したチャンクの差分データをレプリケーションにより冗長化することで、データ更新時のパリティ更新を不要とし、EC 化による性能低下を抑止する。差分データ管理の詳細は 4.2 節で後述する。

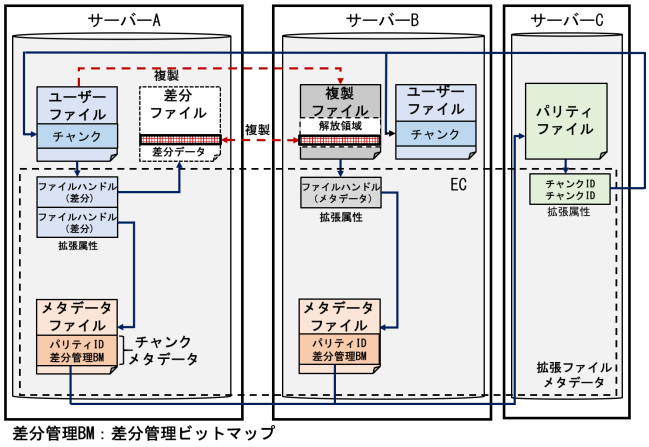

図 3　データレイアウト

一方、ページ単位更新監視はチャンク内の更新をローカルファイルシステムのページ単位で監視することで、更新偏りの有無を判定可能とする。DRC は、従来のチャンク単位のアクセス監視に加え、ページ単位の更新有無を監視し、更新箇所をオンメモリの更新ビットマップにより管理する。DRC は EC 化の対象とするチャンクの選択時に、更新偏りの大きさを調べ、更新偏りが大きい場合は差分データ管理付きの EC 化対象とする。従来方式に対し、EC 化できる容量が増えるため、より多くの冗長データ容量を削減することが可能となる。ページ単位更新監視の詳細は 4.3 節で後述する。

これら 2 つの機能を組み合わせることで、従来方式に対して更なる容量効率改善が可能となる。

4.2　差分データ管理

提案方式では、DRC は更新偏りがあるチャンクを EC 化した際に、そのチャンクに対する差分データ管理を開始する。DRC は差分データ管理のため、差分データを格納するための特殊ファイルである差分ファイルと、ローカルファイルシステムのファイルメタデータを拡張した拡張ファイルメタデータ [13] を使用する。差分ファイル、拡張ファイルメタデータを用いた際の差分データ管理のデータレイアウトを図 3 に示す。

DRC は、ユーザーファイルごとに、差分データ管理を開始するタイミングで、差分ファイルを作成する。差分ファイルの実体は、ローカルファイルシステムに格納された通常ファイルであり、データ実体を持たない空ファイルとして作成される。DRC は、ユーザーファイルの拡張ファイルメタデータに、差分ファイルのファイルハンドルを格納することで、ユーザーファイルから差分ファイルを参照可能とする。

また、DRC は、差分データ管理が有効となっているチャンクに対して更新アクセスがあった際には、更新データを差分ファイルに書き込むことで、EC 化されたデータ更新に伴うパリティ更新を回避する。DRC は、差分データを、レプリケーション・ペアにある複製ファイルの解放領域に複製することで、差分データを冗長化する。差分ファイルはレプリケーションにより保護されているため、EC 化されたチャンク本体を更新する場合に比べ、低コストで更新することができる。

DRC は、チャンクごとに用意した差分管理ビットマップを用いて、ページごとの差分データの有無を管理する。DRC は元々、EC を構成するチャンクとパリティの識別子を、拡張ファイルメタデータに相互に格納することで、サーバ—間の EC を実現していた。差分データ管理では、差分ビットマップを拡張ファイルメタデータの新たな要素として格納することで、差分データ管理の構成情報を管理する。DRC は、差分データ管理を行うチャンクを EC 化した後に、該当チャンクに更新が有った場合、差分管理ビットマップの対応するフラグをオンにし、差分データを差分ファイルに格納する。

DRC は、これらの差分データを該当ページの更新がある限り差分ファイルに保持し、一定期間更新がなかった場合に、ユーザーファイルに反映した後に差分データを解放する。

4.3　ページ単位の更新監視による更新偏り検出

DRC は、チャンクに対するアクセスをモニタし、モニタデータを元に EC 化するチャンクを選択する。提案方式では、DRC は従来のチャンクごとの IO 数に加え、ページ単位の更新有無を監視対象とする。DRC は、ページ単位の更新有無を、メモリ上の更新ビットマップに記録し、チャンクに対する更新偏りの大きさを判断するために使用する。

従来方式では、チャンクの EC 化の優先度を式 (1) を用いて計算する。

$$EC_{priority} = \frac{1}{1 + IO_{write} + \alpha IO_{read}} \qquad (1)$$

ここで $EC_{priority}$ は EC 化する際のチャンクごとの優先度を表す。IO_{read} と IO_{write} はそれぞれ一定期間内にあったチャンクに対する参照アクセス数と更新アクセス数を表す。α はリード重みづけのための係数である。正常時には EC 化の性能低下は更新時のみに発生するため、α はデフォルト 0 とし、障害時の性能保証が必要な場合は 1 を使用する。DRC は上記優先度に従い、使用容量に対する EC 化したデータの割合が、事前に指定された EC 化閾値 R に達するまで EC 化する。ユーザは、性能要件を満たすために必要な EC 化するデータへのアクセス比率をもとに R の値を決めることで、性能と容量効率のトレードオフを設定する。

一方、提案方式では上記の優先度に従い EC 化したチャンクに加え、更新偏りの大きいチャンクも差分データ管理付きの EC 化対象とする。DRC は、優先度に基づく EC 化を行ったのち、残りのチャンクに対して更新偏りの大きさを判断し、更新偏りの大きなチャンクを差分データ管理付きで EC 化する。更新偏りの大きさは、更新ビットマップを用いて判断し、差分データ管理に必要な容量が、EC 化により削減可能な容量より小さいかどうかで判定する。これらの判定は、更新ビットマップからチャンクの差分管理に必要な差分データ容量の比率 D を式 (2) から見積もり、式 (3) から求まる EC 化による容量削減率 E と比較することで行う。

$$\text{差分データ容量率 } D = \frac{\text{更新ありのページ数}}{\text{チャンク内のページ数}} \qquad (2)$$

表 1　評 価 環 境

設定値	値
マシン	HP Proliant DL160 G6
CPU	Intel(R) Xeon E5620 2.40GHz、4 コア
メモリ	12GB
ディスク	HP 120 GB 3G SATA 2.5 SFF MDL SSD x 2
NIC	HP NC550SFP 10 GbE Server Adapter
OS	Ubuntu 16.04.4 (Linux 5.0.1 kernel)
ベンチマーク	filebench1.5-alpha3

$$\text{容量削減率 } E = (m–k)/m \qquad (3)$$

ここで m はデータシンボル数、k はパリティ数をそれぞれ表す。

DRC は、更新データとその複製分を合わせた $2D$ が E よりも小さいチャンクについては、EC 化による容量削減が見込めるため、更新偏りが大きいと判定し、差分データ管理有りで EC 化する。

このように、更新偏りが大きなチャンクを、差分管理付きの EC 化対象とすることで、従来方式に比べてより高い容量効率を実現することができる。

5　評　　価

本章では提案方式の有効性を確認するため、プロトタイプによるアクセス性能評価と、シミュレーションによる容量削減効果の評価を実施する。

5.1　アクセス性能

5.1.1　評 価 環 境

アクセス性能評価では、汎用サーバー（HP Proliant DL160) を 3 台を使用し、そのうち 1 台でベンチマークツールの filebench [14] を動作させた。各サーバー上で DRC を動作させ、レプリケーションで冗長化した構成に対する、従来の差分データ管理なしで EC 化した構成 [8] と、提案方式の差分データ管理ありで EC 化した構成の、それぞれについて性能比較を行った。ここでいうレプリケーションした構成とは全チャンクをレプリケーションした構成、EC 化した構成とは全チャンクを 2D+1P にて EC 化した構成である。ワークロードは 5 多重の 64KB シーケンシャルリード/ライト、1 多重の 8KB ランダムリード/ライト、および 1 多重のログライト、10 多重の DB ライト、200 多重の DB リードからなる OLTP を用いた。データセットサイズは、メモリ容量の 2 倍とした。アクセス性能の評価環境を表 1 に示す。

5.1.2　評 価 結 果

評価結果を図 4 に示す。図中では、各ワークロードごとにレプリケーションに対する性能比を、従来方式である差分データ管理なしの EC、提案方式である差分データ管理ありの EC のそれぞれについて示している。

シーケンシャルリード (Seq Read)、ランダムリード (Rand Read) は、レプリケーションに対し、従来方式、提案方式共に同等性能となっている。これは、DRC ではローカルファイルシステムに格納したユーザファイルをサーバ間をまたがって EC 化するため、EC 化したチャンクへのリードはローカルアクセスになるためである。

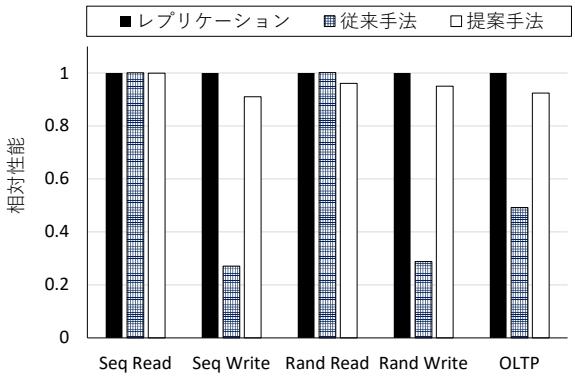

図 4　アクセス性能

一方、ライトを伴うシーケンシャルライト（Seq Write)、ランダムライト (Rand Write)、OLTP は、提案方式の性能が従来方式に対し大幅に上回っている。従来方式がレプリケーションに対し、シーケンシャルライトで 73%、ランダムライトで 71%、OLTP で 51%性能が低下するのに対し、提案方式はそれぞれ 9%、5%、8%の性能低下にとどまっている。その結果、提案方式は従来方式に対し最大 236%のアクセス性能の改善を実現している。

これらの結果は、提案方式が差分データ管理を行うことにより、EC による性能低下を回避できていることを示している。

5.2　容 量 効 率

提案方式による容量削減効果を確認するため、シミュレーションによる容量削減率の見積もりを行った。評価は、TPC-C ベンチマークを PostgreSQL に対して実行した際の IO トレースをもとに、提案方式による容量削減量を計算した。IO トレースは、TPC-C ベンチマークにおいて更新主体のアクセスパターンとなる stock テーブルを取得対象とした。また、性能要件に対する容量削減の違いを評価するため、全更新アクセス数のうち EC 化したチャンクへの更新アクセス比率が 10%以下となる容量だけ EC 化した場合と、30%以下となる容量だけ EC 化した場合の 2 通りについて評価した。更に、EC の構成の違いによる影響をみるため、EC の構成は 2D+1P、6D+2P、13D+2P の 3 通りを見積もった。評価結果を図 5 に示す。

グラフではオリジナルのユーザーデータ容量に対する削減可能な容量の比を、従来方式と提案方式それぞれについて各構成に対して示している。ここでいう削減可能な容量とは、ユーザーデータの複製データをパリティデータに置き換えることにより、削減できる冗長データの量を指している。また、提案方式については EC 化により削減可能な容量に加え、EC 化した

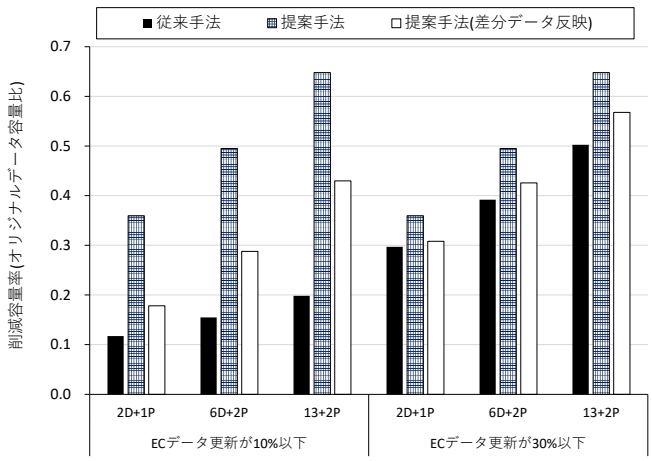

図 5　TPC-C stock テーブルにおける容量削減効果

後に更新により発生する差分データ量を加味した容量削減率も示している（「提案方式（差分データ量反映）」）。差分データの見積もりでは、IO トレースに現れる更新アクセスが、EC 化した後にも再びを繰り返し更新されることを想定している。

グラフからわかる通り、EC 化したデータへの更新アクセス比率を同等とした場合、提案方式が従来方式に対してより多くの冗長データを削減できている。提案方式による容量削減効果は、EC 化したデータへのアクセス比率を 10%とした場合に、従来方式の最大 3.2 倍、アクセス比率を 30%とした場合に最大 1.29 倍となっている。これは、提案方式では更新偏りが大きいチャンクを差分データ管理付きで EC 化することで、EC 化するデータ容量を増やすことができるためである。

また、差分データ量を加味した場合でも、提案方式の容量削減量は従来方式を上回っている。差分データ量を加味した提案方式の容量削減効果は、EC データへのアクセス比率を 10%とした場合に従来方式の最大 2.17 倍、アクセス比率を 30%とした場合に最大 1.13 倍となっている。

これらの結果から、提案方式を用いることで、OLTP のような細粒度の更新偏りがあるワークロードにおいて、従来方式に対して容量効率を改善できることがわかる。

6　関 連 研 究

本研究が対象とした、アクセス監視によるホットデータとコールドデータの識別は、冗長化制御に限らず多くのストレージ関連技術に使用されている。

階層ストレージの分野において、アクセス監視は階層移動の対象とするデータ選択のために広く使用されている [15] [16]。これらの階層ストレージにおいても、冗長化制御と同様に、細粒度の更新偏りが発生した場合に、上位階層の使用効率低下が発生する。著者ら知る限り、この課題に対して解決策は示されていない。本研究の提案方式は、階層ストレージにおいても上位階層の使用効率改善に有効であると考えられる。

インメモリ DB の分野では、アプリケーション側のアクセス監視をもとにホットデータとコールドデータを見極め、データ格納先を切り替える方式が提案されている。Lvandoski [17] や Stoica [18] らは、インメモリ DB のストレージ制御がタプルのアクセス頻度に応じてデータの格納先をオンメモリとセカンダリストレージ間で切り替える制御を提案している。これらのアプリケーション側のアクセス監視では、アプリケーションの管理データ単位でのアクセス監視が可能となり、提案方式のようなストレージ側での監視に比べ、より精度の高いコールドデータの検出が可能となる。一方で、これらの手法ではアプリケーションごとに独自のアクセス監視が必要となり、汎用性の点で不利となる。提案方式は、ストレージ側のアクセス監視を前提とすることで、アプリケーションに非依存な汎用的な容量削減手法を実現している。

7　今後の課題

本研究のアクセス性能評価では、全データをレプリケーションした構成と EC 化した構成のそれぞれについて比較評価した。一方、実運用においては、性能要件に応じて EC 化するデータの割合は異なり、また障害時の性能についても考慮する必要がある。EC 化する割合を変えた場合の性能や、障害時の性能評価が今後の課題である。また、本研究では、提案方式による容量削減効果を TPC-C ベンチマークを用いたシミュレーションにより検証した。今後は、より多様なワークロードに対して提案方式が有効であることを確かめることが課題となる。

8　お わ り に

本研究ではサーバーベースストレージにおける、ワークロードを考慮した容量効率改善手法を提案した。提案方式は、従来方式に対し、差分データ管理と更新偏りを考慮した EC 化制御を導入することで、性能を維持した容量効率改善を実現した。評価では、提案方式を用いることにより EC 化したデータに対するアクセス性能を、従来方式に対して最大 236%改善することを示した。また、シミュレーションにより、提案方式が TPC-C ベンチマーク実行時の容量削減効果を、従来方式に対して最大 2.17 倍まで改善することを確認した。これらの結果から、提案方式によりサーバーベースストレージの更なるコスト削減が可能となることを示した。

文　　献

[1] Steve Mattos. Server-based storage. In *Flash Memory Summit 2012*, 2012.

[2] Luiz Andre Barroso, Urs Holzle, Parthasarathy Ranganathan, and Margaret Martonosi. *The Datacenter As a Computer: Designing Warehouse-scale Machines*. Morgan & Claypool Publishers, 3rd edition, 2018.

[3] Philipp Reisner and Lars Ellenberg. Replicated storage with shared disk semantics. 2007.

[4] Microsft. Storage replica overview, 2019. visited on 2019-11-30.

[5] Vikhyat Umrao, Michael Hackett, and Karan Singh. *Ceph cookbook: practical recipes to design, implement, operate, and manage Ceph storage systems*. Packt Publishing Ltd, 2017.

[6] Mohammed Rafi KC. Efficient data tiering in glusterfs. In *Storage Developer Conference (SDC) 2016*, 2016.

[7] James S Plank. Erasure codes for storage systems: A brief primer. *Login: The USENIX Magzine*, Vol. 38, No. 6, pp. 44–50, 2013.

[8] Takayuki Fukatani, Hieu Hanh Le, and Haruo Yokota. Lightweight dynamic redundancy control for server-based storage. In *Proceedings of the 38th International Symposium on Reliable Distributed Systems*, pp. 353–369. IEEE, 2019.

[9] Takayuki Fukatani, Hieu Hanh Le, and Haruo Yokota. サーバーベースストレージにおける更新頻度による動的冗長化制御手法. 第１１回データ工学と情報マネジメントに関するフォーラム論文集, 2019.

[10] Yue Yang and Jianwen Zhu. Write skew and zipf distribution: evidence and implications. *ACM transactions on Storage (TOS)*, Vol. 12, No. 4, p. 21, 2016.

[11] Wenguang Wang and Richard B Bunt. Reference behaviour of the tpc-c benchmark. 2000.

[12] PostgreSQL.org/. Postgresql: The world's most advanced open source relational database, 2019. visited on 2019-11-30.

[13] Takayuki Fukatani, Atsushi Sutoh, and Takahiro Nakano. A method to adapt storage protocol stack using custom file metadata to commodity linux servers. *International Journal of Smart Computing and Artificial Intelligence*, Vol. 2, No. 1, pp. 23–42, 2018.

[14] Vasily Tarasov, Erez Zadok, and Spencer Shepler. Filebench: A flexible framework for file system benchmarking. *login: The USENIX Magazine*, Vol. 41, No. 1, pp. 6–12, 2016.

[15] 相坂勇気, Le Hieu Hanh, 横田治夫. デバイスミックスストレージシステムの深層強化学習を用いたデータ再配置. 第１１回データ工学と情報マネジメントに関するフォーラム論文集, 2019.

[16] Junpeng Niu, Jun Xu, and Lihua Xie. Hybrid storage systems: a survey of architectures and algorithms. *IEEE Access*, Vol. 6, pp. 13385–13406, 2018.

[17] Justin J Levandoski, Per-Åke Larson, and Radu Stoica. Identifying hot and cold data in main-memory databases. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 26–37. IEEE, 2013.

[18] Radu Stoica and Anastasia Ailamaki. Enabling efficient os paging for main-memory oltp databases. In *Proceedings of the Ninth International Workshop on Data Management on New Hardware*, p. 7. ACM, 2013.