社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

データベース再編成機能を有するストレージシステムの構築

合田　和生†　　喜連川　優†

† 東京大学 生産技術研究所 〒 153-8505 東京都目黒区駒場 4-6-1
E-mail: †{kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし　本論文は，データベース再編成機能を有する高機能ディスクアレイである SRS(自己再編成ストレージ) を提案する．SRS はデータベース再編成をその内部でオンラインに実施することにより，データベースの構造の効率性を保つことが可能である．ディスクアレイ内の豊富な IO 帯域と高い IO 処理能力を有効に利用するため，並列パイプラインデータ処理，物理アドレスレベル IO 制御，独自の高速ログ適用処理を実施する．商用 DBMS を対象とした試作機を実装し，性能評価実験を行い，サーバ上で実行される従来的なデータベース再編成に比べ，性能の大幅な改善が可能であることを示す．

キーワード　データベース再編成，高機能ストレージシステム，データベース管理，性能チューニング，構造劣化

Introducing Database Reorganization into Storage Systems

Kazuo GODA† and Masaru KITSUREGAWA†

† Institute of Industrial Science, The University of Tokyo, Komaba 4-6-1, Meguro-ku, Tokyo, Japan 153-8505
E-mail: †{kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract　This paper proposes *SRS (Self-Reorganizing Storage)*, a highly functional disk array which has the capability of database reorganization. SRS conducts database reorganization online in itself and keeps the structural efficiency independently of DBMS running on server systems. Parallel pipelined data processing, physical IO scheduling and log processing are adopted in order to take full advantage of high internal bandwidth and IO processing power of disk storage systems. An SRS prototype is implemented using a commercial DBMS. The experimental results reveal that significant performance improvement is achieved.

Key words　Database Reorganization, Highly Functional Storage Systems, Database Administration, Structural Deterioration

1. はじめに

記憶装置上のデータは，更新によってその構造が劣化する恐れがある．構造劣化は，データアクセスの性能を著しく悪化させる可能性がある．ストレージ上で永続的に管理されるデータベースにおいては，この問題は深刻である．故に，ストレージ上のデータを再配置することにより，劣化した構造を回復し性能を改善する再編成はデータベースに不可欠な機能である．

オフライン再編成は最も単純な再編成の方式である．再編成はデータの再配置のために膨大な IO を要するため，負荷は極めて大きく長時間の処理を必要とする．このためデータベースの構造劣化と再編成のコストを慎重に見極めて，再編成の判断をすることがデータベース管理者には求められる．

一方，近年データベースには極めて高い可用性が要請されており，再編成処理を行う時間的猶予は十分与えられていないことから，サービス継続中に再編成を可能なオンライン再編成 [7] が強く求められている．オンライン再編成ではトランザクション処理と再編成が競合するため，副作用を最小化することが求められると同時に，再編成を効率良く高速に完了することも必要となる．このため，同時実行性の向上，IO の効率化に焦点を当てた実行方式に関する研究が行われ，高度なオンライン再編成機能が利用されるようになりつつある [8]- [17]．しかし，再編成は最も IO 指向の処理の一つである一方，今日の一般的なシステム構成ではサーバ・ストレージ間の IO 帯域も十分ではなく，サーバ装置の IO 処理能力には限界がある．故に，IO 処理の競合は避けられない為，副作用を十分に防ぐことはできず，また同時に高速化も困難であるという問題が存在する．

本論文ではデータベースのオンライン再編成を効率化することを目的として，再編成機能を有する高機能ディスクストレージである自己再編成ストレージ (Self-Reorganizing Storage; SRS) システムを提案する．SRS はオンラインでデータベースの再編成を実施し，記憶空間の構造の効率性を維持する．ス

トレージ装置内で再編成を実施することにより，サーバ・ストレージ間の IO 帯域やサーバ装置の IO 処理における競合は回避され，トランザクション処理性能への副作用を最小化することができる．ストレージ装置が有する高い IO 処理能力と豊富な IO 帯域を活用し，IO 指向の処理方式を導入することにより，再編成を極めて高速に実施することが期待される．近年においてはデバイス技術の進展から二次記憶装置は急速に高機能化しており，既に多くの IO 指向のデータ管理機能 [19]- [21] がストレージシステム内で実施されるようになりつつあり，情報システムの複雑性解消に寄与している．このような技術の進展を鑑みるに，サーバ上の DBMS がトランザクション実行を行う一方，ストレージがデータの物理管理を実施することは，自然な解法である．本論文の提案である SRS は再編成タスクを DBMS から切り離すことを可能とする点において，情報システムの設計の容易化に貢献すると期待される．

本論文は，上記の提案に基づき，SRS の設計を述べる．また，有効性を検証するために，商用 DBMS である HiRDB [4] を対象とした試作機を実装し，ベンチマークアプリケーションを用いた性能評価を示す．なお，我々はオープンソース DBMS である MySQL [6] を対象とした試作機においても実験を行い，同様の効果を得たが，その有効性は紙面の都合から別稿に譲る．

本論文の構成は以下の通りである．2. では関連研究を紹介する．3. では SRS の基本設計を述べ，さらに，再配置とログ追い付きを高速化する詳細設計を解説する．4. では商用 DBMS を対象とした SRS の試作機の実装を紹介し，試作機による TPC-H 及び TPC-C ベンチマークを用いた性能評価実験を紹介し，有効性を論じる．5. では本論文をまとめる．

2. 関 連 研 究

これまでのデータベースのオンライン再編成はサーバ上で実行されるソフトウェアによって実施され，その方式は主に同時実行再編成 (Concurrrent Reorganization) と分離再編成 (Split Reorganization) の 2 つに分類することができる．

同時実行再編成は，ユーザのトランザクションと再編成を同時実行する．本方式では特に競合の解消に関する研究が鍵である．以下の 2 つの研究はこれらの問題について，詳細な議論を行っている．

C.Zou と B.Salzberg らは, [9] において行移動トランザクションに基づく同時実行再編成を提案している．行を移動した場合，行を参照する索引を更新する必要があるが，当該トランザクションが依存性を解決できることを詳細に述べている．また，トランザクション間のスケジューリングを行うことにより，問い合わせ性能への副作用を低減する試みがなされている. [10] では主鍵索引の B+木に関して，専用のロックプロトコルを設けることにより実行並列度を向上させる手法が述べられている. [11] では二次索引の更新を主記憶上のサイドファイルに記録し，バッファマネージャがユーザトランザクションの IO 要求に合わせてピギーバックして更新することにより，更新 IO 数を削減する手法が述べられている. [10] [11] では再編成にかかる処理のログを取得することにより，再編成中の障害に対しても安全に復旧することができることを詳細に述べている．

E.Omiecinski らは，クラスタリングされたレコードについて [12] [13] では小粒度のロックを用いた交換に基づく同時実行再編成を述べている．この際，バッファの利用を効率化し IO コストを最小化する問題を，超グラフと代表グラフを用いて定義し，それが巡回セールスマン問題に帰着可能であることを示す．発見的な解法を示し，バッファのページングアルゴリズムへと統合する. [14] では索引の更新をサイドファイルを用い索引の更新 IO を削減することを示し，提案手法の解析モデルに基づきシミュレーションによる性能評価を示す．

その他，同時実行再編成に関する IO 制御方式に焦点を当てた研究が存在する．P.Zabback らは [15] において，IO キューを監視しディスク負荷が低い時に再編成 IO を発行することにより，再編成の副作用を低減する IO スティールなる手法を提案している．また，E.Thereska らは [16] において同様に，再編成にディスクの空き時間をメインテナンス作業に用いるフリーブロックスケジューリング [18] を利用することを提案している．S.Ghandeharizadeh らは [17] において，メディアサーバにおけるアプリケーションの IO にストライピング再構成の IO をピギーバックさせる手法を提案している

分離再編成は，先ず再編成対象空間の複製空間を作成し，複製空間の再配置を行い，その後，再配置中に行われた更新操作を複製空間に反映し，最後に空間の切替えを行う．空間の有効性を犠牲にすることにより，排他競合を解消することが特徴である．多くの商用 DBMS におけるオンライン再編成機構は分離再編成を採用している [1] [2] [3] [5] [8]．

G.Sockut らは [8] においてファジー再編成とログ追い付きによる分離再編成を提案している．著者の知る限り，当該論文は分離再編成の実行方式に関して詳細に記述した唯一の論文である．再編成対象空間からファジーダンピングによるコピー作成を行い，コピー空間で再編成を実施し，さらに更新を反映するためログをコピー空間に適用する手法を提案している．コピー空間にログを適用するにあたり，ログの参照アドレスを変換し，変換後のアドレスによって整列することにより，IO 効率を高める．また，ファジーダンピングによって生成されるコピーは一貫性が保証されないが，後のログ追い付きによって安全に回復可能であることが詳細に述べられている．

3. 自己再編成ストレージ

3.1 システムアーキテクチャ

本節では，SRS(自立再編成ストレージ) の基本設計を述べる．図 1 に SRS のシステム構成を示す．通常，高機能ディスクアレイは，広帯域内部スイッチ，ディスクモジュール，メモリモジュール，制御モジュールから構成される．制御モジュールは複数のプロセッサ，IO ポート，メッセージ用のポート等から構成され，RAID 等の仮想化制御，キャッシュ制御を司る．メモリモジュールは他のモジュールから共有され，主にキャッシュデータや仮想化の変換表の保持などに利用される．SRS では，再編成のためのソフトウェアを制御モジュール上で実行することが可能であり，特に，再編成のためのソフトウェアが動作す

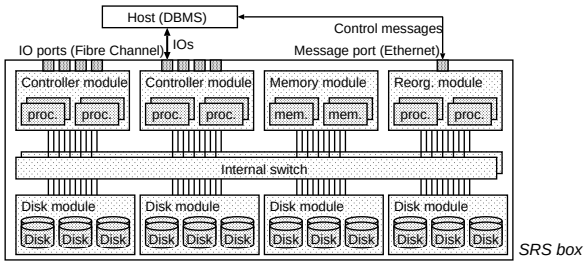


図 1　SRS system architecture.

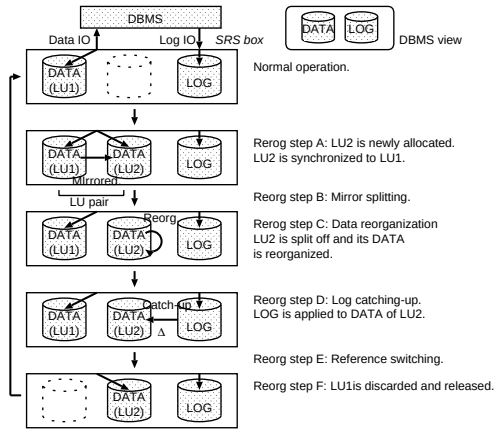


図 2　Reorganization steps.

る制御モジュールを再編成モジュールと呼ぶ．SRS では，ディスクアレイ内の制御モジュールにおける RAID 等の機能によってサーバに提示される論理ディスクである LU(Logical Unit) をデータベースの表空間として利用することを前提とし，分離再編成に基づくオンライン再編成を実施する．

再編成は再編成命令を通信インターフェースを通じて再編成モジュールに与えることにより開始する．再編成命令は再編成の対象となる表空間と，再編成に関する設定パラメータ (目標充填率など) を指定する．再編成命令が再編成モジュールに到着すると，再編成モジュールは図 2 に示す以下の手順により再編成を実施する．

A.　**複製の作成**　　表空間として利用されている LU(以後 LU1 と呼ぶ) に対して，同じ構成の LU(以後 LU2 と呼ぶ) を動的に作成し，LU1 にミラーさせる．即ち，LU1 上のデータは LU2 に複製され，LU2 は LU1 の同期複製となる．

B.　**ミラー対の分離**　　LU 間の同期が確立した後，SRS は再編成対象となる表空間を処理している DBMS に静止化を要求する．DBMS は静止化要求に従い，当該表空間に関連する新規トランザクションの受付を停止し，実行中のトランザクションが終了するのを待ち，バッファのフラッシュを行う．静止化が完了した後，SRS は RAID のマッピングを書き換え当該表空間を構成する LU 対のミラーを切り離し，DBMS のアクセスが LU1 を参照するように変更し，再度 DBMS へ静止化の解除を要求する．DBMS は当該表空間の静止化を解消し，新規トランザクションの受付が開始される．

C.　**データの再配置**　　SRS は LU2 に対して，データの物理配置を変更することにより，再編成を実施する．この際，表の行及び索引のエントリの移動履歴を記述した行 ID 変換表 (RCT) 及び索引エントリ ID 変換表 (ECT) が作成される．この処理の間，DBMS は LU1 を参照して，ユーザのトランザクション処理を実行する．RCT は (旧 ROWID→ 新 ROWID) のリスト，ECT は (旧 ENTID→ 新 ENTID) のリストである(注1)．

D.　**ログ追い付き**　　C. 実行中に DBMS のアクセスにより LU1 は更新されているため，SRS はその間に記録されたログを LU2 へ適用することにより，LU2 を LU1 へ論理的に追い付かせる．この際，DBMS のログは LU1 のアドレス空間に対して記述されているため，RCT 及び ECT を用いることにより，当該ログを LU2 に適用可能なように変換する．

E.　**参照 LU の切替え**　　SRS は再び DBMS に静止化を要求する．DBMS は静止化要求に従い，静止化を実施する．静止化の完了後，RAID のマッピングを書き換え DBMS のアクセスが LU2 を参照するように切替え，再度 DBMS へ静止化の解除を要求する．DBMS は当該表空間の静止化を解消し，新規トランザクションの受付が開始される．

F.　**複製の解放**　　SRS は LU1 のデータを破棄し，LU1 を解放する．

A., B., E., F. は DBMS の有する静止化機能及び，ストレージ装置が有する RAID 機能や仮想化機能，PiT コピー [19] 作成機能を用いて実現することができるため，本論文では詳細を述べない．一方，C. および D. の手続きはストレージの新しい機能であり，実質的にこの手続きを高速化することが再編成の高速化につながるため，以降の節においてその詳細を述べる．

E. に於いてログ領域へ DBMS のログ書き込みと SRS のログ読み込みが競合するが，一般に，データと比較してログ書き込みが性能に与える影響は小さく，両者の負荷ともシーケンシャルであることからディスクアレイのキャッシュにより書き込みへの副作用はないものと考えられる．これ以外の点では，DBMS のトランザクション処理と SRS の再編成は物理的に分離されているため，再編成による問い合わせ性能への副作用はないものと考えられる．

図 3 に SRS においてデータの再配置及び後述のログ追い付きを行うソフトウェア構成を示す．再編成モジュール内では移動ユニット，整列ユニット，ログ適用ユニット，バルク挿入ユニット，IO ユニット及び管理ユニットがが配置される．管理ユニットは常に 1 つ存在し，IO ユニットはディスクの数存在する．その他のユニットは，プロセッサ数毎に起動可能である．メモリモジュール上の共有メモリ空間にはページ，ディレクトリ情報，RCT，ECT，及びログウィンドウのバッファを設ける．

3.2　データ再配置の高速化

索引は表に対して ROWID による依存性を有するため，再編成後の索引エントリが適切に再編成後の行を参照できるように再配置を実施する必要がある．SRS では再編成対象の表空間の組に対し，先ず，ディレクトリ情報，再編成命令の中の充填率

(注1)：一般に多くのデータベースでは表の行及び索引のエントリは，ページ ID とページ内部のスロット番号の組等により物理的にアドレス参照される．本論文ではこの組を ROWID(行 ID)，ENTID(索引エントリ ID) と呼ぶ．

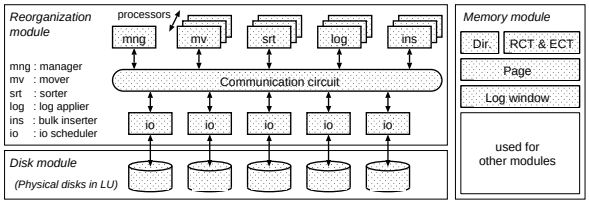
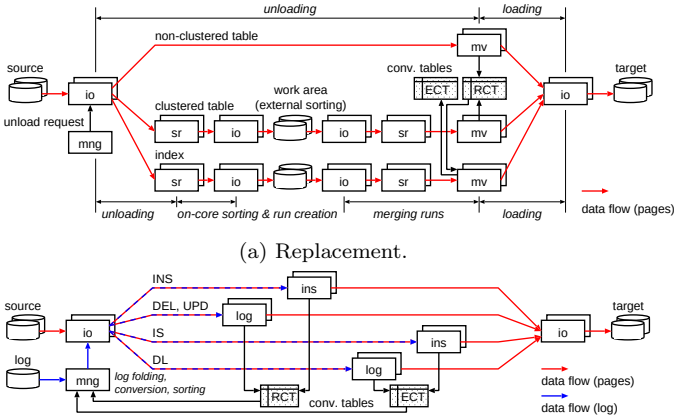


図 3　Software structure.



(a) Replacement.

(b) Log catching-up.

図 4　Parallel pipelined data processing.

目標を元に，アンロード後の必要空間量を見積り，データの再配置計画を立て，空間を確保する．その後，全ての表ページを再配置を実施し，行の移動を記述する RCT を構築する．最後に，全ての索引の葉ページの再配置を実施し，同時に RCT による ROWID 変換を実施し，エントリの移動を記述する ECT を構築する．その後，再配置された葉ページを元に，枝ページ，根ページに関しては再構成される．

上記の手続きの高速化を図って，SRS はストレージ内部の豊富な IO 帯域と高い IO 処理能力を有効に利用するため，並列パイプラインデータ処理と物理アドレスレベル IO スケジューリングによる高スループット処理を実施する．

SRS は図 4(a) に示すデータ処理モデルを以って，アンロード，整列，ロードを並列にパイプライン処理して，データの再配置を行う．非クラスタ表に関しては，再編成元空間からのアンロードと再編成先空間へのロードがパイプライン化により同時実行され，1 回の走査で再編成が完了する．クラスタ表及び索引に関しては，再編成元空間からアンロード，主記憶上でのクイック整列による整列(注2)及び一時領域への書き込みがパイプライン化により同時実行され，その後，一時領域からのマージ整列による読み込みと再編成先空間へのロードがパイプライン化により同時実行され，2 回の走査で再編成が完了する．

SRS ではディスク毎のスケジューリングを実施することにより，IO 並列度を高め，高速化を図る．図 5 に IO ユニットにおける物理アドレスレベル IO 制御を図示する．各 IO ユニットは割り当てられたディスクへの IO のみを担当し，4 つのスケジューリング待ちのためのページ IO 要求キュー (IORQ) と 1

(注2)：ラン長は利用可能な主記憶上のバッファ長に依存する．

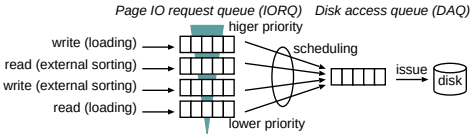


図 5　Physical IO scheduling in IO unit.

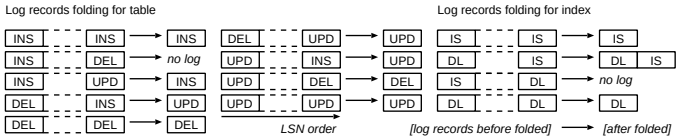


図 6　Log folding.

つのディスクアクセス待ちのディスクアクセスキュー (DAQ) を用いて IO アクセスを行う．ページ IO は読み込み (アンロード)，書き込み (整列)，読み込み (整列)，書き込み (ロード) の順でより高位の優先度を有し，各優先度に IORQ が対応する．IORQ では以下のスケジューリングが実施される．第一に，シーケンシャリティを高めるため，ページ IO 要求は LBA 順に整列される．第二に，アクセスのオーバヘッドを削減するため，アクセス先の隣接したページ IO 要求は一つの IO 要求にまとめられる．IO ユニットは，DAQ 内に要求がある場合，これをディスクに発行する．DAQ が空になると，上述のスケジューリングが行われ，優先度に基づき DAQ は IORQ に接続され，要求が渡される．応答性能よりスループットを重視するため，DAQ は新たな要求の到着により再スケジュールされることはない(注3)．以上の手続きにより，SRS はパイプライン処理のデータ処理モデルとディスク毎に IO 制御を行うことにより，ディスクアレイ内帯域の効率的な利用を図る．

3.3　ログ追い付きの高速化

データ再配置中に記録されたログを再編成後の LU2 に適用することにより，論理的に LU2 を LU1 に追い付かせる必要がある．しかし，ログは LU1 に対して記録されており，行や索引エントリを ROWID や ENTID を以って参照するため，直接 LU2 に適用することはできない．このため，SRS は RCT 及び ECT を用いてログを LU2 に適用可能なように変換する．

ログ追い付きを高速に実施するために，ログウィンドウバッファを設け，ログ畳み込み及びログ整列を実施する．ログ畳み込みは同一の行及び同一行を参照する索引エントリに対する更新ログを集約することにより，適用するべきログレコードの数を削減する．ログ畳み込みは図 6 に示す集約パターンを以って，ログ変換に先だって実施される．ログ変換の後，ログ中のログレコードは新たな ROWID 若しくは ENTID によって整列され，適用される．このログ整列により，ログ適用時の IO の連続性を高めることが可能となる．

高スループットで実施することを目指し，ログ追い付き処理は図 4(b) に示す並列パイプラインデータ処理モデルで実施さ

(注3)：簡単のため，以上の手続きは LU はパリティ無しのストライピング構成されていると仮定しているが，RAID5 等のパリティ付きストライピングにも容易に応用可能である．

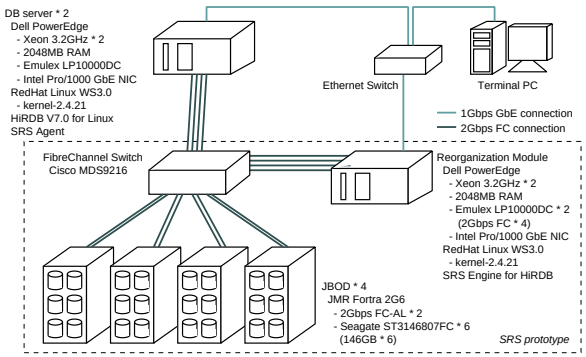
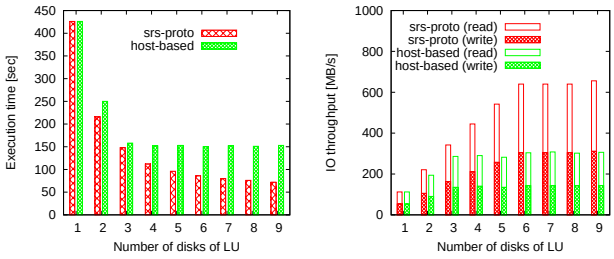


図 7　Experimental system.



(a) Execution time.　(b) IO throughput.

図 8　Disk parallelism effect on reorganization performance.

れる．この際，IO ユニットにおけるスケジューリング制御は前節で述べた方式を利用することにより，読み込みと書き込みの競合を回避し，IO シーケンシャリティを高め，同時に高い IO 並列度を図る．

4.　試作機の実装と性能評価

4.1　試作機の実装

性能評価実験のため，商用 DBMS である HiRDB [4] を対象とした SRS 試作機を実装し，更にデータベースサーバを接続した実験システムを構築した．実験システムの概要を図 7 に示す．SRS 試作機は内部スイッチとしてのファイバチャネルスイッチ，24 個のディスク群としての JBOD，及び再編成モジュールとしての PC サーバにより構成される．当該 PC サーバは 2 個の 3.2GHz Xeon プロセッサ，2048MB 主記憶を有する．IO アクセス用に 4 本の 2Gbps ファイバチャネルによりファイバチャネルスイッチと接続されるとともに，制御通信用にギガビットイーサネットワークにも接続される．さらに，Linux オペレーティングシステム上で，再編成モジュールの実装であるソフトウェア (SRS エンジン) が動作する．データベースサーバ上では HiRDB が動作すると共に，再編成モジュールとの連携を管理する専用のデーモンソフトウェア (SRS エージェント) が動作し，ギガビットイーサネットにより SRS エンジンと通信する．また，専用の PC 端末がデータベースサーバ及び再編成モジュールを管理する．管理端末はデータベースサーバ，再編成モジュールから情報を収集し，ユーザが設定した条件により，再編成モジュールに再編成命令を発行する機能を有する．

当該実験システムは再編成モジュールの有効性の検証に焦点を当てているため，ディスクアレイ外部へ LU を提示する RAID 機能の模擬は，SRS 内部では行わず，データベースサーバ側で Linux カーネルの `md` ドライバによるソフトウェア RAID 機能を利用し，LU を模擬する．再編成モジュールは，SRS エンジンの持つマッピング情報をデータベースサーバ側の `md` のマッピング情報に同期させることにより，同じ LU を模擬する．

4.2　データ再配置の性能評価

SRS はパイプライン処理のデータ処理モデルとディスク毎に IO 制御を行うことにより，ディスクアレイ内帯域の効率的な利用を図る．本方式の有効性を検証するため再配置の性能測定を実施する．なお，DBMS の管理するデータベースのうち，ここでは，表データの再配置に焦点を当てる．

TPC-H におけるスケールファクタを 16 とした初期データに対して，仕様によって規定された更新手続き RF1 及び RF2 による更新 (更新率は 1%とする) を 10 回行ったデータを対象に，目標充填率を 100%とした再配置を実施する．SRS の設定は，共有ページバッファ長を 512MB とし，IO 制御のキュー長をディスク毎に 64 とした．この時，LU はストライピング構成とし，ディスク台数を 1 から 9 まで変化させ，再配置の実行時間，IO スループットを測定する．本実験のデータ再配置においては再配置先として新たな LU を構成するものとする．

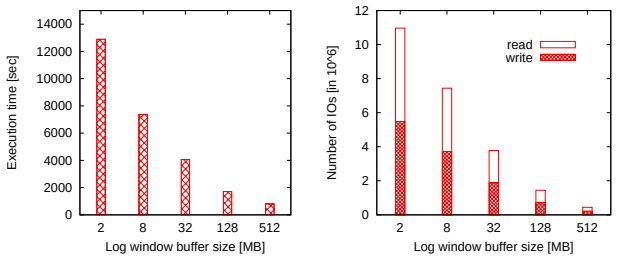
比較対象として，同様の方式によるサーバ上でのオフラインでのアンロード・ロード処理による再編成を測定する．この場合，IO 制御をディスク単位ではなく，カーネル標準の `md` デバイスを用いて LU 単位で行う．サーバ上では `md` デバイスを有効に機能させるため，主記憶の一部をディスクキャッシュとして利用可能とする(注4)．通常，サーバ-ストレージ間の IO 帯域は，ディスクアレイの内部帯域と比較して狭いが，本実験では敢えて方式の比較を行うため，同等の帯域を用意した．

計測結果を図 8 に示す．ディスクの台数が少ない場合，両方の場合で同様の測定結果が得られており，方式間に大きな相違はないと言える．ディスクの台数が 3 台以上の場合，IO スループットの増加が **host-based** の場合では 300MB/s 程度で飽和しており，実行時間の改善が見込めない．一方，**srs-proto** の場合では，IO スループットが最大で 660MB/s まで増加し，実行時間の改善に寄与していることが分かる．**host-based** も LU 単位で IO 制御を実施しており，**srs-proto** に比べ膨大なキャッシュ空間が利用可能であるが，オーバヘッドが無視できない．**srs-proto** ではディスク単位で IO 制御を行い，オーバヘッドの削減を図ることから，より高い IO 並列度を得ることができることが確認できる．本実験結果から，データ再配置においてディスクストレージの IO 制御能力及び高い IO 帯域を利用するためには物理アドレスレベル IO 制御が不可欠であり，提案手法が有効であることが分かる．

4.3　ログ追い付きの性能評価

SRS はログをウィンドウバッファ上でログ畳み込み，及びログ整列を行うことにより，高速なログ追い付きを図る．本方式

(注4)：通常，未使用の主記憶をカーネルのディスクキャッシュとして用い，LRU とエレベーションアルゴリズムに基づく制御を行う．本実験に於いては，およそ 1GB 程度の主記憶がディスクキャッシュとして利用可能であった．



(a) Execution time. (b) IO cost.

図 9 Buffer size effect on catching-up performance.

の有効性を検証するため，ログの追い付きの性能測定を実施する．なお，DBMS の管理するデータベースのうち，ここでは，表データのログ追い付きに焦点を当てる．

本実験では予め一定量のログを生成し，これをログとして適用し，測定する．実際のログ追い付きでは，最後の数回の繰り返し処理において，追い付き判定のため，適用ログのウィンドウ長が小さくなる可能性があるが，これが性能全体に与える影響は小さいため，これを考慮しない．

TPC-C については warehouse 数を 16 とした初期データに対して 100 万トランザクションを実行し，これを再編成の基底状態とする．LU 対を分離し，LU1 に於いて，基底状態に対して，目標充填率を 100%とした再配置を実施し，さらに変換表を作成する．その後，LU2 に於いて，基底状態に対して，100 万トランザクションを実行し，更新ログを得て，これを再配置中の更新操作によって発生したログと見なす．再配置後の表空間に対し，RCT 及び ECT を用いてログの適用を実施する．SRS の設定は，共有ページバッファ長を 512MB とし，IO 制御のキュー長を 64，LU は 3 台のストライピング構成とした．

高速ログ適用処理におけるウィンドウ長を 2MB から 512MB まで変化させ，ログ適用の実行時間を測定する．計測結果を図 9(a) に示す．この結果，ウィンドウ長の増大によって，大幅に実行時間を改善することが可能であることが分かる．さらに分析を行うため，各ウィンドウ長におけるログ適用に必要な論理 IO 数を測定する．図 9(b) に結果を示す．ウィンドウ長の増大により，畳み込み処理によって適用ログの数自体が減少し，同一ページへの更新が集約され，さらに IO の連続性が高まり，IO 数を大幅に低下させることが分かる．以上の実験結果から，ログ畳み込み，ログ整列を用いたログ追い付きが有効に機能し，実行時間を大幅に削減することが可能となることが分かる．

5. ま と め

本論文はデータベース再編成機能を有する高機能ディスクアレイである SRS(自己再編成ストレージ) を提案する．SRS はデータベース再編成をその内部で実施することにより，データベースの構造の効率性を保つことが可能である．ディスクストレージ内の高い IO 処理能力と豊富な IO 帯域を有効に利用するため，並列パイプラインデータ処理，物理アドレスレベル IO 制御，高速ログ適用処理を実施する．商用 DBMS を対象とした試作機を実装し，性能評価実験を行い，実験の結果，従来的なデータベース再編成に比べ，性能の大幅な改善が可能であることが示された．

本論文で提案した再編成のアーキテクチャはストレージシステムにおける高度なデータ管理の基礎となり得ると考えられる．

謝 辞

本研究の一部は，文部科学省リーディングプロジェクト e-Society 基盤ソフトウェアの総合開発「先進的なストレージ技術」の助成により行われた．協力企業である株式会社日立製作所より多くの有益なコメントを頂戴した．感謝する次第である．

文 献

[1] Oracle Database 10g Online Data Reorganization & Redefinition. White Paper. Oracle. 2004.
[2] A.Mohamed, G.Candia, and D.Sherwin. Compareing Architectures of Online Reorganization. White Paper, Quest Software. 2002.
[3] Easing the Pain of an IMS Reorganization. White Paper, BMC Software. 2002.
[4] Hitachi Relational Database Management System Solutions for Disaster Recovery to Support Business Continuity. Review Special Issue, Hitachi Technology. 2004.
[5] Nonstop operation: HiRDB. `http://www.hitachi.co.jp/Prod/comp/soft1/global/prod/hirdb/nonstop.html`. Hitachi Ltd.
[6] MySQL: The World's Most Popular Open Source Database. `http://www.mysql.com`.
[7] (Ed.)D.Lomet. Special Issue on Online Reorganization. IEEE, Bulletin of TCDE, 19(2). 1996.
[8] G.H.Sockut, T.A.Beavin, and C-C.Chang. A Method for On-Line Reorganization of a Database. IBM Systems Journal 36(3). 1997.
[9] B.Salzberg and A.Dimock Principles of Transaction-Based On-Line Reorganization. VLDB. 1992.
[10] C.Zou and B.Salzberg. On-line Reorganization of Sparsely-populated B+-trees. SIGMOD Conference. 1996.
[11] C.Zou and B.Salzberg. Safely and Efficiently Updating References During On-line Reorganization. VLDB. 1998.
[12] E.Omiecinski, and P.Scheuermann. A Global Approach to Record Clustering and File Reorganization. SIGIR. 1984.
[13] E.Omiecinski. Incremental File Reorganization Schemes. VLDB. 1985.
[14] E.Omiecinski, L.Lee, and P.Scheuermann. Performance Analysis of a Concurrent File Reorganization Algorithm for Record Clustering. IEEE Trans. Knowl. Data Eng. 6(2). 1994.
[15] Peter Zabback, Ibrahim Onyksel, Peter Scheuermann, and Gerhard Weikum. Database Reorganization in Parallel Disk Arrays with I/O Service Stealing. IEEE Trans. Knowl. Data Eng. 10(5). 1998.
[16] E.Thereska, J.Schindler, J.Bucky, and Brandon Salmon. A framework for building unobtrusive disk maintenance applications. USENIX FAST. 2004.
[17] S.Ghandeharizadeh, and D.Kim. On-line Reorganization of Data in Scalable Continuous Media Servers. DEXA. 1996.
[18] C.R.Lumb, J.Schindler, G.R.Ganger, D.F.Nagle, and E.Riedel. Towards higer disk head utilization: extracting free bandwidth from busy drives. USENIX OSDI. 2000.
[19] A.Azagury, M.Factor, W.Micka and J.Satran. Point-in-time Copy: Yesterday, Today and Tommorrow. NASA/IEEE MSST. 2002.
[20] EMC Mainframe Solutions Guide. Engineering White Paper. EMC. 2002.
[21] J.Carleton. Electronic Data Archiving and Retrieval in the Public Sector. Analyst Report. Frost & Sullivan. 2004.