データベース再編成機能を有するストレージシステムとその性能評価

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　データベース管理に於いて不可欠な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，即ち自己再編成ストレージ (SRS) と名付けたストレージシステムが著者らにより提案されている．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，データベース内のデータ構造の効率性を保つことが可能であり，情報システムの設計の容易化に貢献すると期待される．また，SRS は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ことを目的として，並列パイプラインデータ処理，物理アドレスレベル IO スケジューリング，及び独自の高速ログ適用処理を実施する．本論文は SRS の設計を述べるとともに，商用 DBMS 及びオープンソース DBMS を対象とした試作機による性能評価実験を示し，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明らかにする．

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，構造劣化

Storage System with Database Reorganization and Its Performance Evaluations

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　The authors have proposed *Self-Reorganizing Storage (SRS)*, 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 high-speed log application mechanism 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 and an open-source DBMS. This paper describes design and prototype implementation of SRS and presents the performance evaluations. The experimental results reveal that significant performance improvement is achieved.

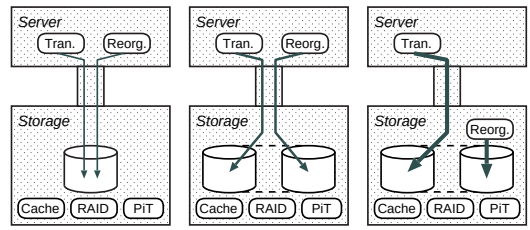
Key words　Database Reorganization, Highly Functional Storage Systems, Database Administration, Structural Deterioration

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り，構造劣化によりデータアクセスの性能は著しく悪化する可能性がある．ストレージ(注1)上で永続的に管理されるデータベースに於いては，この問題は深刻であることから，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．

近年ではデータベースには極めて高い可用性が要請されていることから，サービス継続中に再編成を実行可能なオンライン再編成 [1] に関する研究が行われて来た [2]〜[15]．これまでに提案されているオンライン再編成はサーバ上の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，図 1(a) に示すように，排他制御により DBMS 上に於いてユーザのトランザクションと再編成を同時実行する．一方，分離再編成は，図 1(b) に図示するように，データベースの複製を作成し，元のデータベースではトランザクション処理を，複製データベースでは再編成を実施する．また，再編成終了後に，再編成中のトランザクショ

(注1)：本論文ではストレージを二次記憶装置の意味で用いる．



(a) Concurrent Reorg. (Conventional)　(b) Split Reorg. (Conventional)　(c) Split Reorg. (SRS)

図 1　データベース再編成のアーキテクチャ

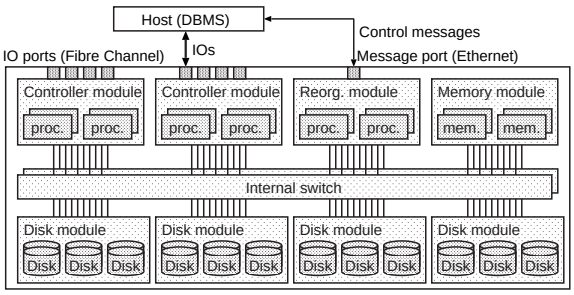


図 2　SRS のシステムアーキテクチャ

ンによる更新を複製データベースに反映する必要がある．いずれの方式に於いても，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求められると同時に，膨大なデータを効率良く高速に処理することが必要となる．しかし，再編成は最も IO 負荷の高い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 帯域は十分ではなく，サーバの IO 処理能力には限界がある．故に，サーバ上のデータベース再編成については，IO 処理の競合は避けられないことから副作用を十分に防ぐことは困難であり，また同時にその高速化も容易ではないという問題が存在する．

一方，ストレージ技術を顧みるに，これまでストレージはブロックの書き込み，読み込みのみを行う単純な記憶装置として取り扱われ，全てのデータ管理はサーバ上で行われていた．しかし，近年に於けるデバイス技術の進展により，高機能ディスクアレイは豊富な演算資源，広大なディスクキャッシュ空間，高い内部通信帯域を有するようになり，これらを利用してストレージ装置内で高度なデータ管理を実施することが可能となりつつある．既にサーバフリーバックアップ，PiT(Point-in-Time) コピー [16]，メインフレーム・オープンシステム間のコード共有 [17]，固定コンテンツ用アーカイブのためのブロック共有 [18] など，多くのデータ管理機能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る．このような高度なデータ管理機能は，サーバとストレージの間に新しいインターフェースを導入し，情報システムの設計の容易化に貢献すると期待される．即ち，ストレージがその格納するデータの物理管理を実施することにより，サーバ上の DBMS はその仕事量が軽減され，より高度な応用に注力することが可能となる．また，一般に近年のストレージは多くのサーバに共有される傾向にあり，ストレージ空間の管理は 1 つのサーバに閉じないため，むしろストレージ側で行うことが自然と考えられる．

以上の技術背景から，本論文では，データベース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 (SRS: Self-Reorganizing Storage) システムを提案する．SRS は図 1(c) に示すように，ストレージ内で分離再編成方式に基づきオンラインでデータベース再編成を実施し，記憶空間の構造の効率性を維持する．SRS は再編成タスクをサーバ上の DBMS から切り離すことを可能とする点に於いて，情報システムの複雑性の軽減に寄与すると思われる．ストレージ装置内で再編成を実施することから，サーバ・ストレージ間の IO 帯域やサーバ装置の IO 処理に於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期待される．ストレージ装置が有する豊富な IO 帯域と高い IO 処理能力を活用し，並列パイプラインデータ処理，物理アドレスレベル IO スケジューリング並びに独自の高速ログ適用処理を導入することにより，再編成を高速に実施することを目指す．

著者らは [19] に於いて，SRS の基本設計を述べている．本論文では，更に有効性を検証するために商用 DBMS である HiRDB [20] 並びにオープンソース DBMS である MySQL [21] 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．

本論文の構成は以下の通りである．2. 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．3. では商用 DBMS 並びにオープンソース DBMS を対象とした SRS の試作機の実装を紹介する．4. では試作機による 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.　自己再編成ストレージ

2.1　システムアーキテクチャ

自己再編成ストレージ (SRS) の基本設計を述べる．図 2 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，高帯域内部スイッチ，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 ではデータベース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，再編成ソフトウェアが動作する制御モジュールを再編成モジュール (Reorganization Module) と呼ぶ．SRS では，ディスクアレイ内の制御モジュールに於ける 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論理ユニット (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する．

SRS に於いては，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，分離再編成に基づくオンライン再編成を開始する．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 3 に示す以下の手順により再編成を実施

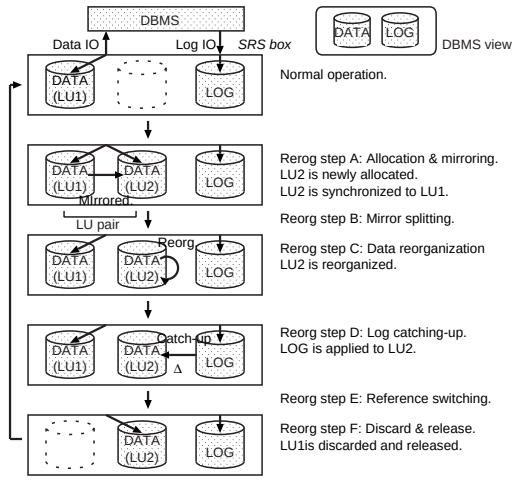


図 3　再編成の手続き

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 (quiescence) 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 のみ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を開始する．これにより LU2 はデータベースレベルでの一貫性のある PiT コピーとなる．

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: RowID Conversion Table) 及び索引エントリ ID 変換表 (ECT: EntryID Conversion Table) を作成する．RCT は (旧行 ID→ 新行 ID) の列，ECT は (旧エントリ ID→ 新エントリ ID) の列である(注2)．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．

D. ログ追い付き　C. データの再配置の実行中に DBMS のトランザクションにより 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，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を開始する．

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 やプロビジョニング等の仮想化機能，PiT コピー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. 及び D. の手続きは本論文が提案する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全体の高速化につながるため，以降の節に於いてその詳細を述べる．

E. に於いてログ領域への 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，表空間のデータと異なり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ログ書き込みへの副作用は無視できるものと想定する．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，再編成によるトランザクション処理性能への副作用はない．

図 4 に SRS に於いてデータの再配置及び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プロセス，整列プロセス，ログ適用プロセス，挿入プロセス，IO プロセス及び管理プロセスが配置される．管理プロセスは常に 1 つ存在し，他のプロセスの管理を行う．IO プロセスはディスク毎に存在し，担当するディスクの IO スケジューリングを行う．その他のプロセスはプロセッサ毎に起動可能であり，データ再配置及びログ追い付きに用いる．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．

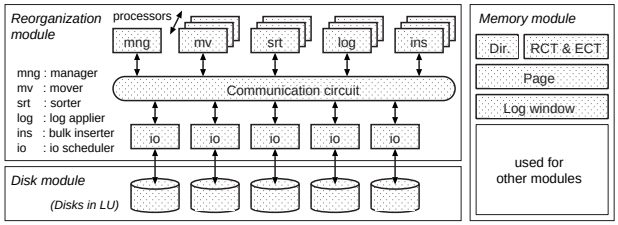
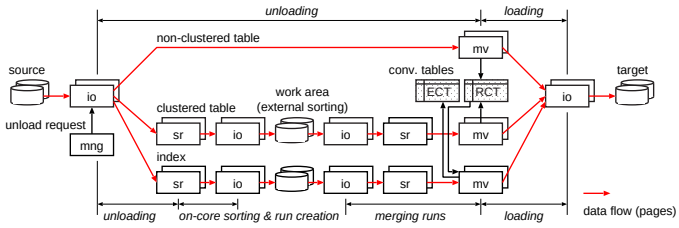


図 4　ソフトウェア構造

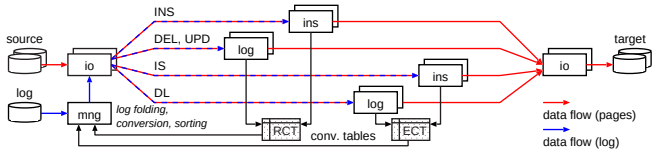
2.2　データ再配置の高速化

索引は表に対して行 ID による参照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，再編成命令中の充填率目標を元に，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，全ての表ページの再配置を実施し，行の移動を記述する RCT を構築する．続いて，全ての索引の葉ページの再配置を実施し，同時に RCT による行 ID 変換を実施し，エントリの移動を記述する ECT を構

(注2)：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表空間内で物理的にアドレス参照される．本論文ではこの組を行 ID，エントリ ID と呼ぶ．



(a) Reorganization.



(b) Log catching-up.

図 5　並列パイプライン化データ処理

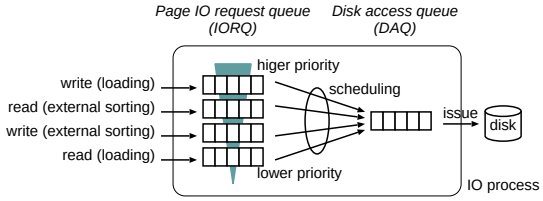


図 6　IO プロセスに於ける物理アドレスレベル IO スケジューリング

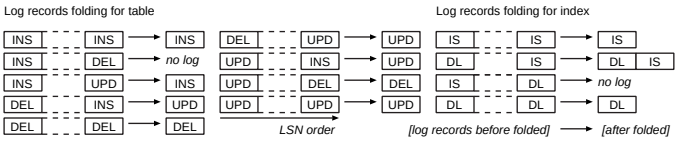


図 7　ログ畳み込み

築する．最後に，再配置された葉ページを元に，枝ページ，根ページを再構築する．

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べく，並列パイプラインデータ処理と物理アドレスレベル IO スケジューリングを実施する．

SRS は図 5(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3)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 ではディスク毎に IO 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 6 に IO プロセスにおける物理アドレスレベル IO 制御を図示する．各 IO プロセス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 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，IO の連続性を高めるため，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 プロセスは，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．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．

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．

2.3　ログ追い付きの高速化

データ再配置の終了後，再配置中に DBMS のトランザクションによる更新によって記録されたログを再編成後の LU2 に適用し，LU2 を論理的に LU1 に追い付かせる必要がある．この際，ログは LU1 に対して記録されており，そのログレコードは LU1 に依存した行 ID 若しくはエントリ ID による参照を有することから，直接 LU2 に適用することはできない．SRS では 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 7 に示す集約規則を以って，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行 ID 若しくはエントリ ID 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 5(b) に示す並列パイプラインデータ処理モデルで実施される．この際，IO プロセス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 の連続性を高め，同時に高い IO 並列度を図る(注4)．

3.　試作機の実装

商用 DBMS である HiRDB [20] 及びオープンソース DBMS である MySQL [21] を用いて SRS 試作機を実装し，更にデー

(注3)：ラン長は利用可能な主記憶上のバッファ長に依存する．

(注4)：ログ追い付きについては，IORQ として高優先度の書き込みキューと低優先度の読み込みキューの 2 種類のみを取り扱う．

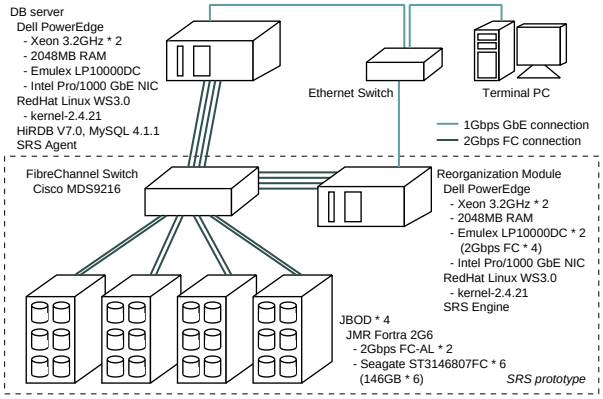


図 8　実験システム

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 8 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，ディスクモジュールとしての 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 Intel Xeon プロセッサ，2048MB の主記憶を有する．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 並びに MySQL 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おり，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データベースサーバから `md` のマッピング情報を取得し，同じ LU を模擬する．

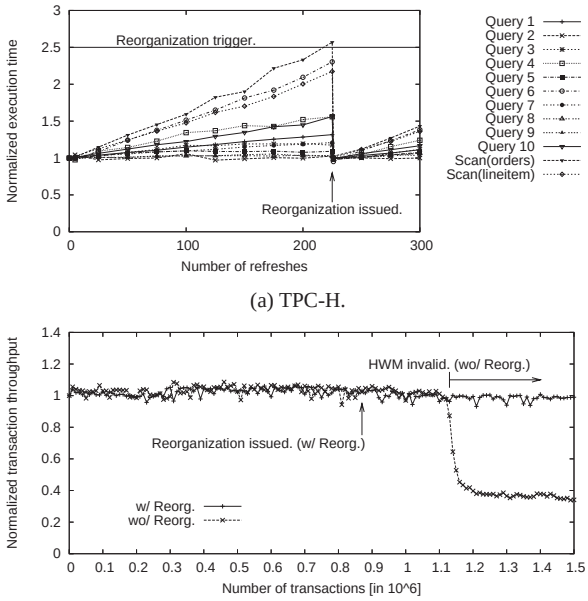
4. 性 能 評 価

SRS の有効性を検証するために，データベースベンチマークである TPC-H 及び TPC-C をアプリケーションとして用い，性能評価実験を行う．本論文では主に HiRDB を用いた試作機の実験に焦点を当てて述べる(注5)．

4.1 構造劣化によるデータベース性能悪化と再編成によるデータベース性能回復

TPC-H 及び TPC-C を用いて，ストレージ上の構造劣化が DBMS の性能悪化に与える影響と，SRS の再編成による性能回

(注5)：MySQL を用いた試作機の実装においても同様の結果を得ているが，その有効性に関しては紙面の都合から一部のみを紹介し，別稿に譲る．



(a) TPC-H.

(b) TPC-C with migration task.

図 9　データベース性能の悪化と回復

復を計測する．この際，HiRDB の設定は全てページ長を 16KB，エクステント長を 32 ページ，共有バッファ長を 512MB とし，ストライピングユニット長を 64KB として 3 台のディスクからストライピング構成される LU を 2 つ作成し，それぞれ表，索引用の表空間として利用する．

TPC-H は典型的な意志決定支援アプリケーションのベンチマークであり，23 個の問い合わせと 2 個の更新操作が定義されている．RF1(New Sales Refresh Function) は新しい売り上げ記録を `ORDERS` 表及び `LINEITEM` 表に追加し，RF2(Old Sales Refresh Function) は古い売り上げ記録を削除する．RF1 と RF2 の組合せによりデータウェアハウスの一部が更新されるが，データウェアハウス全体の大きさは殆ど変化しない．実験は以下の通り行う．非クラスタ化表と索引をスキーマとして定義し，スケールファクタを 8 とした約 8GB の初期データを生成する．RF1 及び RF2 の組合せを 300 回実行することにより徐々にデータウェアハウスを更新し，25 回の更新毎に Q.1 から Q.10 までの問い合わせと，`ORDERLINE` 表及び `LINEITEM` 表の全表検索の応答時間の変化を測定する．RF1 及び RF2 の更新率は 1%とし，DBMS のキャッシュ効果を除くため，問い合わせ毎に共有バッファをクリアする．更に，管理端末は問い合わせの応答時間を監視し，初期状態から 2.5 倍以上に悪化した場合，目標充填率を 100%とした再編成を SRS に命令する．測定の結果を図 9(a) に示す．

問い合わせの内，Q.1, 4, 6, 10 等の `ORDERS` 表若しくは `LINEITEM` 表に対する表走査を実施する処理の応答時間が更新回数に応じて，徐々に悪化することが分かる．これは，行の削除操作に於いて，HiRDB は応答性能を高めるために，該当するページ内のスロットに削除フラグを立て，行データが占めていた空間を回収しないことによる．ページやエクステント全体の行が削除されたとしても，直ちにはページやエクステン

トは開放されないため，全表検索を実施する場合，削除された空間も走査する必要が生じる．当該実験に於いては，RF1 による挿入と RF2 による削除によって，空間の密度が低下し，表走査の効率が低下することによる．また，225 回目の更新の際，`ORDERS` 表の応答時間が 2.5 倍以上悪化したため，SRS により再編成が実施され，問い合わせの応答時間が改善する．以上のことから，SRS の再編成により構造劣化による段階的な性能悪化から性能を回復させることが可能であることがわかる．

一方，TPC-C はオンライントランザクション処理のベンチマークであり，5 種類のトランザクションが定義されている．純正 TPC-C では常に `Order`，`Order-Line`，`History` の各表の行数が単調に増加するが，実際のシステムに於ける長期的な運用の下では，当該表のうち古いデータは定期的にデータウェアハウスやアーカイブなどに移送される．このような長期的なオンライントランザクション処理システムの振舞を模擬するため，定期的に実行されるバッチとして移送処理を導入し，移送処理により `Order`，`Order-Line`，`History` の各表からトランザクションに影響を与えない古い履歴が削除されるようにする．長期的に `Order`，`Order-Line`，`History` の各表の行数をほぼ一定に保つように，移送処理によって削除される行数は適切に制御される．即ち，移送処理が N トランザクション毎に実行される場合，当該移送処理は直前の N トランザクションによって追加された行数とほぼ同数の行を削除する．実験は以下の通り行う．本実験のみ表空間長は 2GB に限定し，非クラスタ化表と索引をスキーマとして定義し，ウェアハウス数を 16 とした約 1.5GB の初期データを生成する．思考時間を 0 として，150 万トランザクションを実行し，1 万トランザクション毎のスループットを計測する．移送処理は 10 万トランザクション毎に起動される．実験は SRS による再編成を実施しない場合 (**wo/Reorg.**) と，実施する場合 (**w/Reorg.**) の測定を行った．**w/Reorg.** の場合，管理端末は後述する HWM の位置を監視し，表空間の 95%以上に到達したことを契機として，目標充填率を 100%とした再編成を SRS に命令する．測定結果を図 9(b) に示す．

実験開始後，いずれの場合もしばらくの期間はスループットが安定的な値を保っているが，**wo/Reorg.** の場合，113 万トランザクション近郊から急激に低下する．これは，HiRDB はオンライントランザクション処理に於ける挿入操作の応答性を重視していることから，非クラスタ化表への挿入操作については，主記憶上に管理される高水準位 (HWM: high-water mark) なるポインタを利用することによる．HWM は各表について割り当てられた最大のページ ID を指しており，挿入操作は HWM を参照することにより，高速な挿入を実施することが可能となる．その半面，HWM が表空間の終端まで到達した場合，HWM は機能せず，挿入操作に対して空きページを検索する必要が生じ，挿入の応答時間が悪化する．当該実験に於いては，113 万トランザクションまでは HWM ポインタを利用して挿入を実施しているが，それ以降は HWM ポインタが表空間の端に到達し，無効化されることによる．即ち，表空間内には空き空間が存在するが，挿入は空き空間を探索する必要があるため，結果的にスループットが低下する．一方，**w/Reorg.** の場合，あらかじめ 88 万トランザクション近郊に於いて，SRS により再編成が実施され，その後の HWM 無効化を防ぐことが可能となっており，スループットの低下を防ぐことができることが分かる．

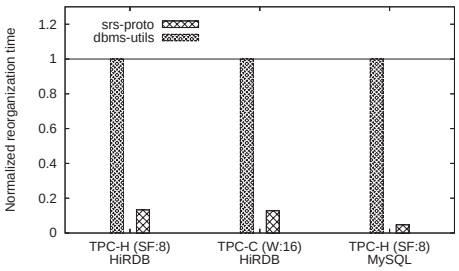


図 10　SRS と DBMS ユーティリティのデータ再配置の性能

以上のことから，SRS の再編成により構造劣化による性能悪化を未然防止することが可能であることがわかる．

4.2　データ再配置の性能評価

SRS はディスクアレイの持つ内部 IO 帯域と IO 処理能力を効果的に活用して高スループットの再編成を実現するため，並列パイプライン化データ処理と物理アドレスレベル IO スケジューリングを行う．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，本実験では表データの再配置に焦点を当てる．

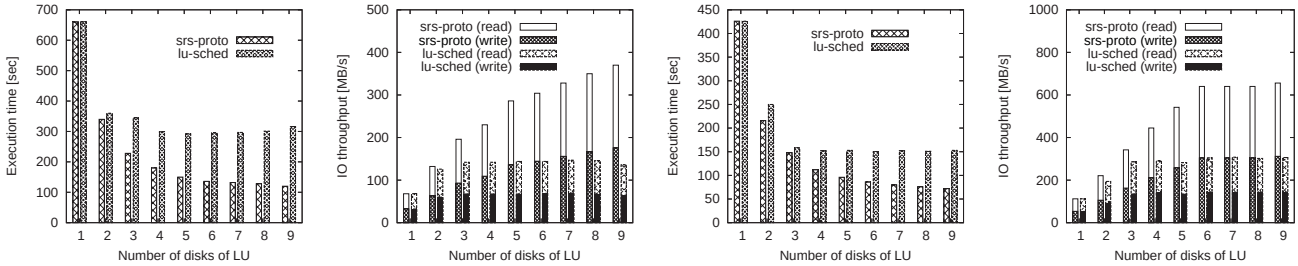
4.2.1　DBMS の再編成ユーティリティとの比較

4.1 と同じ実験環境に於いて，TPC-H 及び TPC-C のデータに対するデータ再配置の実行時間を測定する．TPC-H に関しては，スケールファクタを 8 とした約 8GB の初期データに対して RF1 及び RF2 による更新を 5 回行ったデータを対象とする．TPC-C に関しては，ウェアハウス数を 16 とした約 1.5GB の初期データに対して 100 万トランザクションを実行したデータを対象とする．いずれも再編成の目標充填率を 100%とする．SRS の設定は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，LU は 3 台のディスクからストライピング構成されるものとする．この際，SRS 試作機による LU 間再編成 (**srs-proto**) と，DBMS の再編成ユーティリティによる再編成 (**dbms-utils**) の実行時間を比較する．**dbms-utils** の場合，同じディスク 3 台から構成される LU を一時領域として用いる．また，TPC-H のデータに対する同様の実験を MySQL に於いて行った．

図 10 に計測結果を示す．HiRDB, MySQL に於いて，DBMS の再編成ユーティリティと比較してそれぞれ約 13%，約 5%の実行時間でデータ再配置を完了していることが分かる．このため，今日一般的に用いられている再編成ユーティリティと比較して，SRS の再編成により大幅な性能改善が可能であることが分かる．

4.2.2　物理アドレスレベル IO スケジューリングの効果

ディスクアレイ内の IO 資源が有効に利用されていることを確認するために，前節と同じ実験設定において，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，RF1 及び RF2 による更新を 10 回行ったデータを対象に，目標充填率



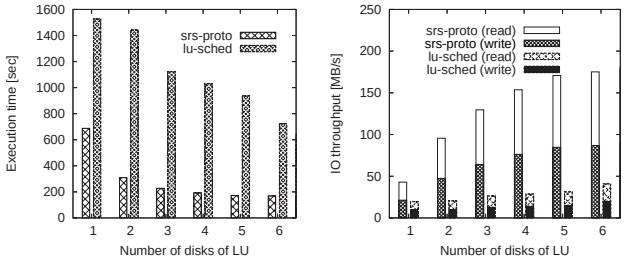
(a) Execution time (intra-LU). (b) IO throughput (intra-LU). (c) Execution time (inter-LU). (d) IO throughput (inter-LU).

図 11　ディスク並列度とデータ再配置性能の関係

を 100%とした再配置を実施し，LU を構成するディスク台数を 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．SRS 試作機の再編成を測定するとともに，比較対象として物理アドレスレベル IO スケジューリングを行わない再編成 (**lu-sched**) を測定する．この場合もデータ処理はパイプライン化して行われる．IO 制御はカーネル標準の `md` デバイスを用いて LU 単位で行い，`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6)．

LU 内でデータ再配置を実施した場合の計測結果を図 11(a)(b) 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 **lu-sched** の場合では約 150MB/s 程度で飽和しており，これ以上の実行時間の改善が見込めない．一方，**srs-proto** の場合では，ディスク 9 台に於いても実行時間，IO スループットが改善しており，最大で約 370MB/s の IO スループットを達成している．**lu-sched** も LU 単位で IO スケジューリングを実施しており，**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ず，スループットの飽和に至っている．一方，LU 間でデータ再配置を実施した場合の計測結果を図 11(c)(d) に示す．LU 内再配置と同様の傾向が得られており，**srs-proto** の場合，最大で 660MB/s という高い IO スループットが得られている．いずれの場合に於いても，ディスク台数が多い場合，**srs-proto** は **lu-sched** の約 2 倍のスループットを達成している．**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，データ再配置に於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 制御が極めて有効であることが分かる．

尚，別の実験環境に於いて MySQL を対象とした実装を用いた場合の，LU 間のデータ再配置の実行時間及び IO スループットの計測結果を図 12 に示す．**lu-sched** に対して **srs-proto** の実行時間は大幅に改善されており，また高い IO スループットが達成されている．本論文の提案する物理アドレスレベル IO スケジューリングが，HiRDB だけでなく MySQL に於いても極めて有効に機能することが分かる．MySQL に於いては表データは B 木に基づくデータ構造に格納され，HiRDB のそれとは大きく構造が異なるが，提案手法が双方に有効に機能することから，その有効性が示されている．



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

図 12　データ再配置性能 (MySQL を対象とした実装)

4.3　ログ追い付きの性能評価

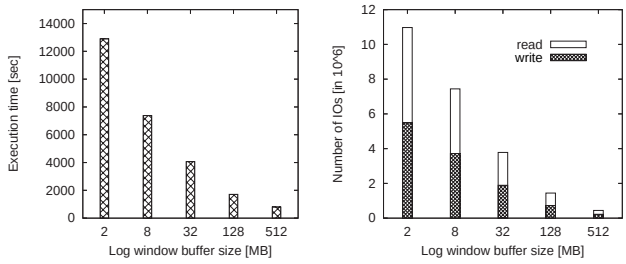
SRS はログをウィンドウバッファ上でログ畳み込み，及びログ整列による処理することにより，高速なログ追い付きを図る．本方式の有効性を検証するため，ログの追い付きの性能測定を実施する．なお，本実験では表データのログ追い付きに焦点を当てる．

本実験では予め一定量のログを生成し，この適用を測定する．4.1 と同じ実験環境に於いて，TPC-C に於けるウェアハウス数を 16 とした約 1.5GB の初期データに対して 100 万トランザクションを実行する．その後，表空間のデータを同じ構成の LU からなる表空間に複製する．元の表空間に於いて更に 100 万トランザクションを実行し，ログを取得する．その後，複製された表空間に於いてデータ再配置を実行し，RCT 及び ECT を生成し，引き続いてログを適用する．この際の SRS の設定は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 台のディスクのストライピング構成とする．

ログ適用処理に於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 13(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．

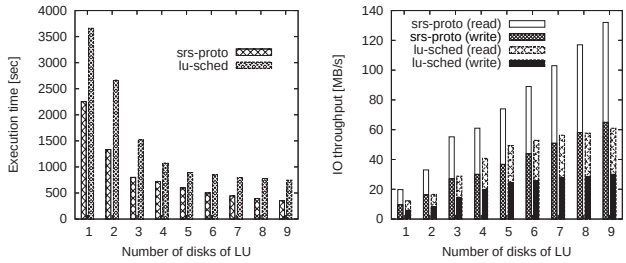
更に，ログウィンドウバッファの効果を分析するため，各ウィンドウ長に於けるログ適用に必要な論理 IO 数を測定する．図 13(b) に結果を示す．ウィンドウ長の増大により，畳み込み処理によって適用ログの数自体が減少すると同時に，ログ整列により IO の連続性が高まり，同一ページへの更新が集約され，IO

(注6)：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 13 ウィンドウ長とログ追い付き性能の関係



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

図 14 ディスク並列度とログ追い付き性能と関係

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．

最後に，ウィンドウ長を 512MB とし，LU を構成するディスク台数を 1 から 9 まで変化させた場合に，実行時間及び IO スループットを計測した．比較のために，**srs-proto** と **lu-sched** 双方の IO スケジューリングによるログ追い付きを測定する．計測結果を図 14 に示す．全ての場合で **srs-proto** が **lu-sched** より優位な結果が得られている．特に，ディスクの台数が 5 台以上の場合，**lu-sched** の場合では IO スループットの増加及び実行時間の改善もが鈍化している．一方，**srs-proto** の場合では，ディスク台数 9 台に於いても IO スループットが増加し，実行時間の改善に寄与していることが分かる．これは，**lu-sched** では LU 単位で IO 制御を実施しているため，ディスク台数の増加に応じた処理の並列度の向上が不十分であるのに対し，**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，より高い並列度を得ることができるためと考えられる．以上の実験結果から，ログ追い付きに於いても本論文の提案する物理アドレスレベル IO スケジューリングが有効であると考えられる．

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自己再編成ストレージ (SRS) を提案した．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 並びにオープンソース DBMS を対象とした試作機を実装し，性能評価実験を行った．実験の結果，提案手法によりデータベース再編成に於けるデータ再配置，ログ追い付き双方に於いて大幅な性能の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於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] (Ed.)D.Lomet. Special Issue on Online Reorganization. IEEE, Bulletin of TCDE, 19(2). 1996.

[2] B.Salzberg and A.Dimock Principles of Transaction-Based On-Line Reorganization. VLDB. 1992.

[3] C.Zou and B.Salzberg. On-line Reorganization of Sparsely-populated B+-trees. SIGMOD Conference. 1996.

[4] C.Zou and B.Salzberg. Safely and Efficiently Updating References During On-line Reorganization. VLDB. 1998.

[5] E.Omiecinski, and P.Scheuermann. A Global Approach to Record Clustering and File Reorganization. SIGIR. 1984.

[6] E.Omiecinski. Incremental File Reorganization Schemes. VLDB. 1985.

[7] 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.

[8] 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.

[9] E.Thereska, J.Schindler, J.Bucky, and Brandon Salmon. A framework for building unobtrusive disk maintenance applications. USENIX FAST. 2004.

[10] S.Ghandeharizadeh, and D.Kim. On-line Reorganization of Data in Scalable Continuous Media Servers. DEXA. 1996.

[11] 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.

[12] Oracle Database 10g Online Data Reorganization & Redefinition. White Paper. Oracle. 2004.

[13] A.Mohamed, G.Candia, and D.Sherwin. Compareing Architectures of Online Reorganization. White Paper, Quest Software. 2002.

[14] Easing the Pain of an IMS Reorganization. White Paper, BMC Software. 2002.

[15] Nonstop operation: HiRDB. `http://www.hitachi.co.jp/Prod/comp/soft1/global/prod/hirdb/nonstop.html`. Hitachi Ltd.

[16] A.Azagury, M.Factor, W.Micka and J.Satran. Point-in-time Copy: Yesterday, Today and Tommorrow. NASA/IEEE MSST. 2002.

[17] EMC Mainframe Solutions Guide. Engineering White Paper. EMC. 2002.

[18] J.Carleton. Electronic Data Archiving and Retrieval in the Public Sector. Analyst Report. Frost & Sullivan. 2004.

[19] 合田和生，喜連川優．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．信学技報 104(537) CPSY2004-58, 2004.

[20] Hitachi Relational Database Management System Solutions for Disaster Recovery to Support Business Continuity. Review Special Issue, Hitachi Technology. 2004.

[21] MySQL: The World's Most Popular Open Source Database. `http://www.mysql.com`.

[22] G.H.Sockut, and Robert P. Goldberg. Database Reorganization - Principles and Practice. ACM Comput. Surv. 11(4). 1979.