一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
DE2018-13 (2018-09)

早期ログ適用技法が再起動時の問合せ実行待ちに与える影響の実験的考察

谷川　祐一†　　合田　和生††　　喜連川　優††

† 東京大学 情報理工学系研究科　〒113–8656 東京都文京区本郷 7–3–1
†† 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: †{tanikawa,kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし　データベース管理システムは，障害等による再起動時に，最新のチェックポイントに遡ってログを適用し，一貫性を確保してから問合せ処理を開始する．早期ログ適用はデータベース管理システムにおいてログ適用と問合せ処理を並行して実行する技法であり，これにより再起動時の問合せ処理の待ち時間短縮が期待される．本研究では当該効果を試作データベースエンジンを用いた実験により明らかにする．
キーワード　データベース管理システム，ログ適用，障害回復

An Experimental Study on Mitigation Effect on Query Execution Suspension by Early Log Application

Yuichi TANIKAWA†, Kazuo GODA††, and Masaru KITSUREGAWA††

† Graduate School of Information Science and Technology, The University of Tokyo　7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–8656 Japan
†† Institute of Industrial Science, The University of Tokyo　4–6–1 Komaba, Meguro-ku, Tokyo, 153–8505 Japan
E-mail: †{tanikawa,kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract　In the failure recovery process, database management systems apply the log from the point-in-time in which the last checkpoint was captured and then start to process given queries on the database in which the consistency is guaranteed. Early log application is a technique to perform log application and query processing concurrently in order to shorten the query suspension time in the recovery process. This paper presents the experiments that we conducted by the use of our prototype database engine to qualify its effect.
Key words　Database Management System, Log Application, Failure Recovery

1. はじめに

データベース管理システムは，大規模データをデータベースとして編成して堅牢に管理し，これに対する問合せ等の処理を迅速に実行するためのソフトウェアである．多くのデータベース管理システムでは，障害等からの回復可能性を実現するために，ライトアヘッドロギングならびにチェックポイントなる制御手法を採用している [1] [2] [3]．ライトアヘッドロギングは，データベースへの更新をログとして書き出すものである．すなわち，データベース管理システムは，更新がログに永続的に記録されたことをもって，更新トランザクションが永続化されたとみなし，これをコミットする．この際，データベースの更新が実際にデータベースを格納するボリュームに永続的に記録されたかどうか，もしくは，データベースバッファ中にダーティーな状態で滞留しているかどうかは問わない．対してチェックポイントは，データベースバッファ中に滞留している更新のうち，コミットされたトランザクションによるものを，データベースを格納するボリュームに永続的に記録するものである．すなわち，チェックポイントが発動された時点において既にコミットされているトランザクションによる更新は，チェックポイントが終了した時点で全てボリュームに永続的に記録されていることになる．

データベース管理システムは，障害等により再起動されると，データベースを格納するボリュームならびにログのディレクトリ情報等を探索し，最新のチェックポイントを同定し，当該チェックポイント以降に記録されたログをデータベースに適用し，データベースを回復し，一貫性のある状態となってから問合せ処理を受け付け，これの処理を実行する．よって，

データベース管理システムに障害等が生じ再起動される場合，データベースを回復するまでの間，データベース管理システムは新たな問合せを受け付けて処理することができない．これはデータベース管理システムの可用性を低下させる 1 つの要因となっている．

早期ログ適用は，データベース管理システムが障害等により再起動された際に，ログの適用によるデータベースの回復を待つことなく，先行的に問合せ処理を実行可能とする技法である [4]．ログ適用中のデータベースは，既にコミットされたトランザクションによる更新が全て適用されていることが保証されない．よって，問合せがデータベースからページを取得する際には，当該ページが既に回復済みであって既にコミットされたトランザクションによる全ての更新が適用されているかどうかを判定し，そうであれば当該ページをそのまま利用し，そうでない場合には当該ページに未適用のログを取得することによりコミットされたトランザクションによる全ての更新が適用されている状態に回復させてから当該ページを利用する．当該技法によれば，データベース管理システムに障害等が生じ再起動される場合，データベースを回復するまで待つことなく，新たな問合せを受け付けて処理することができるようになり，これによってデータベース管理システムの可用性を向上することが期待される．

本論文は，早期ログ適用によってログ適用中に発行された問合せ処理の実行待ち時間が改善する効果を，実験により明らかにする．著者は，早期ログ適用機能を備えたデータベースエンジンを試作し，複数の更新偏りパターンの更新トランザクションによって生成された一連のログを対象として，当該ログの適用中に複数のデータアクセス偏りパターンを有する問合せを発行し，それぞれの組合せで，ログ適用ならびに問合せ処理に要する実行時間や実行中に各種資源利用量を計測し，問合せ処理の実行待ち時間の変化を評価した．当該評価を本論文ではまとめ，評価の結果，早期ログ適用技法を用いることにより，ログ適用中に発行された問合せ処理の実行待ち時間が改善することを明らかにする．

本論文の構成は以下の通りである．2. 章でデータベース管理システムにおける回復可能性と再起動過程について述べ，3. 章で早期ログ適用技法を用いた再起動過程について述べる．続く 4. 章で実験用に試作したデータベースエンジンの実装について述べ，5. 章で実験設定と結果について述べる．最後に 6. 章でまとめと今後の課題について述べる．

2. データベース管理システムにおける回復可能性と再起動過程

2.1 データベース管理システムの回復可能性

多くのデータベース管理システムは，ライトアヘッドロギングとチェックポイントによって障害からの回復可能性を実現している．

ライトアヘッドロギングは，データベースへの更新を実際の更新に先駆けてログとして書き出すもので，ログが永続的に記録されたことをもって，トランザクションによる更新が永続化されたとみなす．チェックポイントは，永続化されたトランザクションの更新を遅れてデータベースに書き込むもので，チェックポイントの終了時点においてトランザクションによる更新がデータベースに永続的に記録されていることになる．

図 1 はデータベースへの更新がディスクに永続化される様子を表した概念図である．横軸を時間として，トランザクションによる更新が，データベースボリュームとログボリュームに記録される時間の様子を矢印として表している．ライトアヘッドロギングによって，トランザクションによる更新はログボリュームには即座に記録されるが，データベースボリュームに実際に記録されるのが保証されるのは次のチェックポイントの終了時点になる．

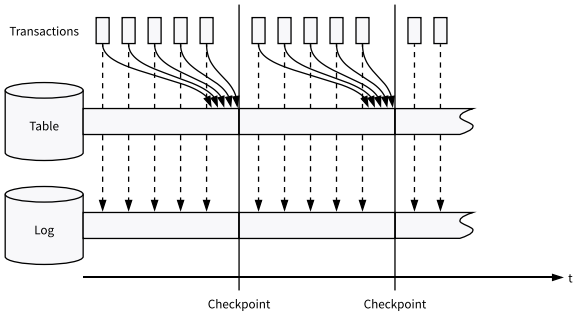

図 1　ライトアヘッドロギングとチェックポイントによるトランザクションの永続化

2.2 データベース管理システムの再起動過程

データベース管理システムの障害からの再起動時には，データベースボリュームから最新のチェックポイントを探し出し，それに対してチェックポイント以後のログを適用することで，データベースを一貫性のある状態に回復する．このとき一貫性のある状態に回復してから初めて問合せ処理の実行を開始する．よってデータベース管理システムの再起動時には，問合せ処理の実行を開始するまでにログの適用待ち時間が存在し，これが可用性低下の一要因になっている．

図 2 はデータベース管理システムの障害からの再起動過程の様子を表した概念図である．大きなバツ印の時点で障害が起こったする．障害時点で最新のチェックポイント以後の更新はログボリュームには記録されているが，データベースボリュームに記録されていることは保証されていない．再起動時には，データベースボリュームの最新チェックポイントに対して，それ以後のログを適用することで障害時点の状態に回復する．このとき，回復のためのログ適用中は問合せ処理を実行することができず，ログ適用の完了を待つ必要がある．

3. 早期ログ適用技法を用いた再起動過程

早期ログ適用技法は，データベース管理システムの再起動時にログ適用によるデータベースの回復を待つことなく，問合せ処理を実行可能にする仕組みである．問合せ処理がデータベースからレコードを読み出すときに，必要に応じて未適用のログを取得してそれを適用したレコードを利用して問合せを

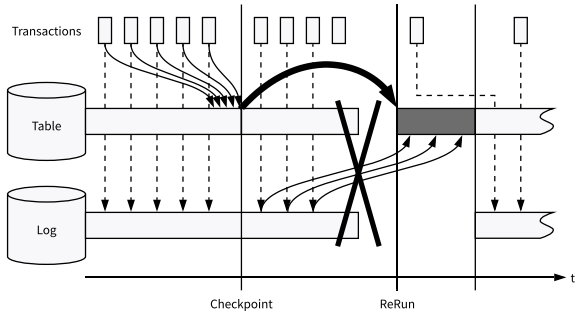

図 2　データベース管理システムの再起動過程

処理する．これによってログ適用中においても，ログ適用後の一貫性のあるデータベースに対する問合せ応答を実現できる．

図 3 は早期ログ適用技法を用いたデータベース管理システムの障害からの再起動過程の様子を表した概念図である．従来の再起動過程である図 2 と比較すると，従来では存在した再起動時のログ適用待ち時間が存在せず，問合せ処理を即座に実行できる．

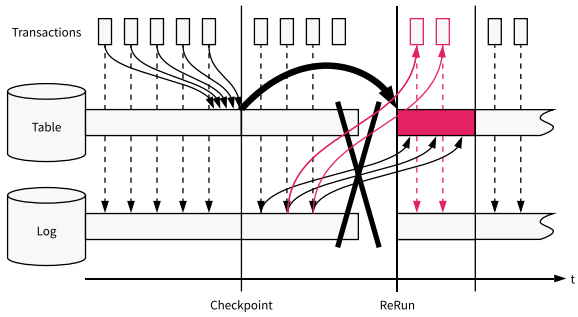

図 3　早期ログ適用技法を用いたデータベース管理システムの再起動過程

具体的には，早期ログ適用技法は，以下の手順を行う．まず，ログ適用中のデータベースは，既にコミットされたトランザクションによる更新が全て適用されていることが保証されない．よって，問合せがデータベースからページを取得する際には，当該ページが既に回復済みであって既にコミットされたトランザクションによる全ての更新が適用されているかどうかを判定し，そうであれば当該ページをそのまま利用し，そうでない場合には当該ページに未適用のログを取得することによりコミットされたトランザクションによる全ての更新が適用されている状態に回復させてから当該ページを利用する．このような手順によって，データベース管理システムに障害等が生じ再起動される場合，データベースを回復するまで待つことなく，新たな問合せを受け付けて処理することができるようになり，これによって，データベース管理システムの可用性を向上することが期待される．

この際，早期ログ適用技法の適用によって，どれほど問合せ実行待ち時間が軽減するのかを実験によって定量的に把握することが，本論文の目的である．具体的には，ログ適用を実行しながら問合せ処理を行った場合に，早期ログ適用を行った場合と行わない場合を比較し，早期ログ適用による問合せ実行待ち時間の軽減効果を把握する．この際，ログが生成された順序でそのまま適用することに加え，ログをアクセス先アドレスの順序に並び替えて適用することによるログ適用効果を検証する．早期ログ適用を行わず整列ログ適用によって単にログ適用にかかる時間を短縮することによる副次的な問合せ実行待ち時間の軽減効果と，早期ログ適用による問合せ実行待ち時間の軽減効果を比較することにより，早期ログ適用による優位性を明らかにする．続く 4. 章以降で，これらの実験の詳細を述べる．

4.　実験用データベースエンジンの実装

早期ログ適用技法による効果を実験によって検証するために，早期ログ適用技法を備えたデータベースエンジンを試作した．本章では試作したデータベースエンジンの機能とその実装について述べる．

4.1　試作データベースエンジンの機能と動作

試作データベースエンジンは単純ログ適用機能，単純問合せ処理機能，早期ログ適用を用いた問合せ処理機能の 3 つの機能を備える．試作データベースエンジンのコンポーネントの概略図を図 4 に示す．

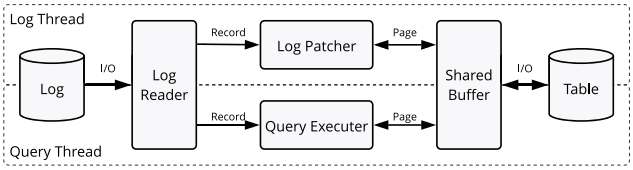

図 4　試作データベースエンジンのコンポーネント概略図

4.1.1　単純ログ適用機能

1 つめの機能は，ログ適用を単体を実行する単純ログ適用機能である．動作としては，ログボリュームからログを読み出して，データベースに適用するというものである．

4.1.2　単純問合せ処理機能

2 つめは，問合せ処理を単体で実行する単純問合せ処理機能である．動作としては，データベースからレコードを読み出して，問合せを処理するというものである．

4.1.3　早期ログ適用を用いた問合せ処理機能

3 つめは，早期ログ適用技法によってログ適用と並行して実行する問合せ処理機能である．この時の問合せ処理の動作は，まずデータベースからレコードを読み出す．その後当該レコードが既に回復済みであるかどうかを判断し，そうであれば当該レコードをそのまま利用し，そうでない場合にはログボリュームから未適用のログを読み出して適用して，問合せを処理するというものである．

従来技術における，データベース管理システムの再起動過程を模擬する時には，単純ログ適用を実行した後に単純問合せ処理を実行する．早期ログ適用技法を用いた再起動過程を模擬する時には，ログ適用中にレコードが回復済みであるかどうかを判断するために，まずログからログ適用の対象となるレコー

ドの索引を作成する．その後，単純ログ適用を実行しつつ，その完了を待たずに早期ログ適用を用いた問合せを実行する．

4.2 試作データベースエンジンの実装の詳細

4.2.1 共有バッファ

ログ適用スレッドと問合せ処理スレッドにおいて共有されるページバッファには，バッファ管理アルゴリズムとして Least Recent Used を採用した．ページ単位の排他ロックを用いて，ログ適用と問合せ実行の並列実行時の排他制御を行う．データベースボリュームへの I/O には pread・pwrite システムコールを用いた．バッファプールのサイズ，すなわちバッファ中に保持するページ数は可変としたが，バッファプールのサイズを変えた時の性能の変化を見るのは今回の実験の目的からは外れているので，実験においてはバッファプールのサイズは一律 128 ページとした．

4.2.2 ログリーダ・ログパッチャ

ログリーダはログボリュームから一度に全てログを読み込み，ログパッチャはログリーダが読み込んだレコードを順番にデータベースボリュームへと適用する．ログを適用する順番として，逐次ログ適用と整列ログ適用の 2 通りのログ適用手法を用意した．

a） 逐次ログ適用

ログが生成された順番そのまま，すなわちログリーダがログを読み込んだ順番そのままに，ログを適用する．

b） 整列ログ適用

ログを，適用先レコードの，データベースボリューム中のテーブルにおけるアドレス順に適用する手法である．これはログ適用における I/O の連続性を高める改善手法で，商法システムなどで利用されている．ログリーダがログボリュームからログを読み込んだ後に，ログを主記憶上でテーブルのアドレス順に並べ替え，ログパッチャがログ適用を行う．

早期ログ適用技法を用いる場合，問合せ処理の対象となるレコードを更新対象をとするログが，ログボリューム中に存在するかどうかを判定する必要がある．そのため，ログリーダはログの読み込み後（整列ログ適用の場合は，ログの並べ替え後），ログの適用先レコードの索引を作成する．

4.2.3 問合せ実行器

問合せ実行器は，任意の SQL による問合せ処理を実行するものではなく，テーブルに対するアクセスパターンと，テーブル中の全レコードに対する読み出すレコードの割合である選択率を指定して，問合せ処理を実行する．

4.3 データベースとログの物理構造

4.3.1 データベースボリューム

データベースボリュームは単一のテーブルで構成される．ページは 8KB，レコードは固定長で 12B とした．索引は省き，一意なレコード ID（一意なページ ID とページ内オフセットの組）によって，レコードとそのレコードを含むページにアクセスする．

4.3.2 ログボリューム

ログレコードは固定長とし，その種類にはデータベースレコードの挿入・更新・削除を想定したが，実験においては更新ログのみを扱うこととした．更新ログは UNDO と REDO のために，更新前と更新後のデータベースレコードをログレコード中に含む．

5. 早期ログ適用技法による問合せ実行待ちの軽減効果の評価

試作したデータベースエンジンを用いてデータベース管理システムの再起動過程を模擬し，再起動時の問合せ実行待ちについて実験を行った．本章では実験環境，実験設定，実験結果について述べる．

5.1 実 験 環 境

実験を行った計算機のハードウェア・ソフトウェア環境を表 1 に示す．データベースエンジンのプロセスは，numactl を利用して単一のスレッドにのみ割り当てた．また発行する I/O については，ダイレクト I/O を利用してシステムのページキャッシュをバイパスさせた．

表 1 実 験 環 境

CPU	Intel Xeon E5-2690 v2 @ 3.00GHz
# of sockets，cores，threads	2，20，40
Memory	64GB
OS	CentOS 6.9（Final）
Kernel	2.6.32-696.20.1.el6
HDD	900GB（10000RPM SAS）
Filesystem	XFS
テーブル用ストレージ	HDD x8 RAID 6
ログ用ストレージ	HDD x2 RAID 1

5.2 実 験 設 定

実験におけるデータセットと負荷の概要を表 2 に示す．

表 2 データセットと負荷

初期データベース	
サイズ	単一テーブル 100GB（6.7 十億レコード）
テストログ	
サイズ	4.6MB（100 千レコード）
種類	更新のみ
生成パターン	Rand，80/20，90/10，99/1，99.9/0.1
適用順序	逐次ログ適用，整列ログ適用
テスト問合せ	
種類	選択のみ
アクセスパターン	Seq，Rand

5.2.1 テストログ

テストログの種類は，データベースレコードの更新のみとした．

ログを生成するパターンには，テーブルへのランダムアクセスを想定して，偏りのないランダムアクセスと，いくつかの偏りの異なるランダムアクセスを用意した．偏りのないランダムアクセスを Rand と表し，これはテーブルの全レコードを無作為に選んで更新した場合とみなすことができる．他方，偏りのあるランダムアクセスを X/Y と表し，これはログの内 X%

のアクセスがテーブルの全レコードの内 Y% を無作為に選んで更新した場合とみなすことができる．この X/Y のパターンとしては，偏りの異なる 80/20，90/10，99/1，99.9/0.1 の 4 通りを用意した．

ログの適用順序は，逐次ログ適用と整列ログ適用の 2 通りを用いた．

5.2.2 テスト問合せ

問合せはデータベースレコードの選択のみとし，選択率 σ（テーブル全体の行に対して選択する行の割合）とテーブルに対するアクセスパターンを変えて問合せを実行した．

アクセスパターンは，シーケンシャルアクセス（Seq）と，ランダムアクセス（Rand）の 2 種類を用いた．なおここでのランダムアクセスは，テーブルの全レコードの内ランダムに選択したレコードを，テーブルの先頭から順にアクセスするものとし，選択率が 1 に近づくほどシーケンシャルアクセスに近いアクセスパターンとなる．

5.3 データベース管理システムの再起動時の問合せ実行待ち

データベース管理システムの再起動過程を模擬し，再起動中に問合せ処理の受付を開始した時の問合せ実行待ちについて計測を行った．

5.3.1 再起動時の問合せ実行待ち時間

テストログとして Rand，テスト問合せとして Rand・選択率 $\sigma = 1e-5$ を用い，再起動と同時に問合せ処理の受付を開始した場合における，問合せ実行待ち時間を図 5 に示す．

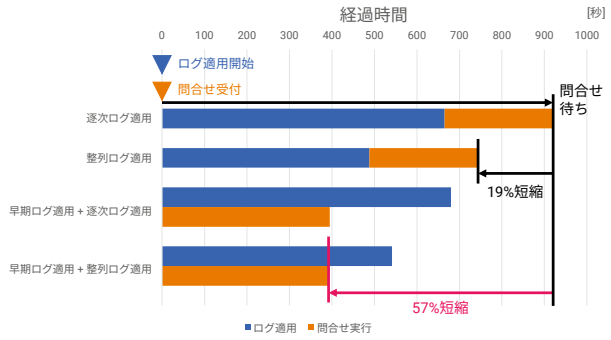

図 5　再起動時の問合せ実行待ち時間

横軸が再起動からの経過時間を表し，ログ適用を実行した時間を青色で，問合せ処理を実行した時間をオレンジ色で表している．上 2 つが従来技術，すなわちログ適用の完了後に問合せ処理を実行した場合で，青色のログ適用の右にオレンジ色の問合せ実行が位置していて，ログ適用の完了後に問合せ処理を実行している様子を表している．下 2 つが早期ログ適用技法を用いた場合で，青色のログ適用とオレンジ色の問合せ処理が上下に重なっていることが，ログ適用と問合せ処理が並列に動作している様子を表している．

問合せ実行待ち時間，すなわち問合せ実行の完了時間を見ると，まずログの適用順序を逐次ログ適用から整列ログ適用にすることで 19% の短縮となり，加えて早期ログ適用技法を用いることで 57% の短縮となった．早期ログ適用技法を用いることによるログ適用にかかる時間へのオーバーヘッドは，逐次ログ適用の場合で 2%，整列ログ適用の場合で 11% となった．

5.3.2 再起動時のリソース利用の変動

再起動過程におけるログ適用と問合せ処理実行時の毎秒の CPU 利用率と I/O スループットの図 6 に示す．5.3.1 節の実験設定のうちの逐次ログ適用の場合を一例として挙げ，左図が従来の再起動過程のもの，右図が早期ログ適用技法を用いた再起動過程のものを表している．

下側の I/O スループットの変動を見ると，従来の再起動過程においては，ログ適用の完了後に読み込み I/O が発行される問合せ処理が実行されることが見て取れる．早期ログ適用技法を用いた再起動過程においては，問合せを受け付けた直後にログ適用と並行して問合せ処理が実行されることが見て取れる．

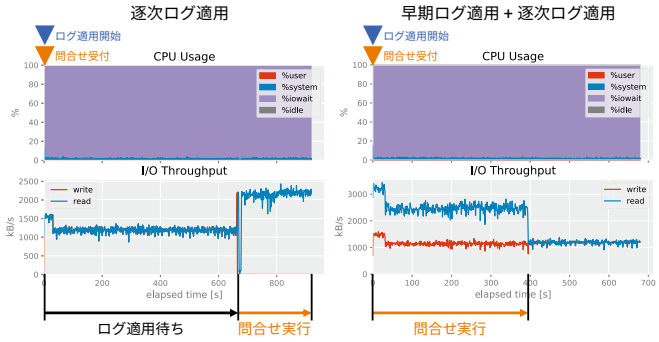

図 6　再起動時のリソース利用の変動（再起動と同時に問合せ受付）

図 7 は図 6 から，再起動の 50 秒後に問合せ処理の受付を開始したという点だけ設定を変えた場合のリソース利用の変動を表している．同様に，従来の再起動過程においては，ログ適用の完了後に問合せ処理が実行される様子が見て取れ，早期ログ適用技法を用いた再起動過程においては，問合せ受付の直後から問合せ処理が実行されることが見て取れる．

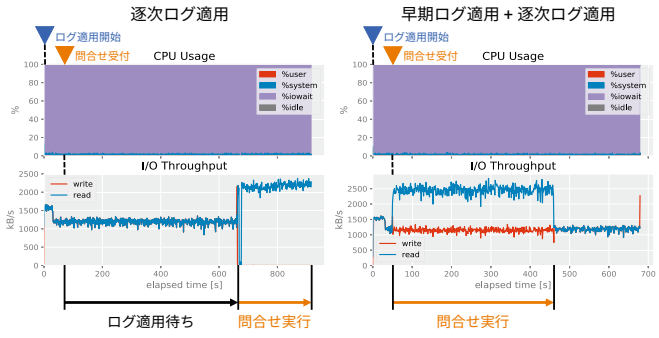

図 7　再起動時のリソース利用の変動（再起動の 50 秒後に問合せ受付）

5.3.3 実験結果のまとめ

整列ログ適用は，逐次ログ適用に比べてデータベース管理システム再起動時の問合せ実行待ち時間を 19% 短縮した．早期ログ適用技法を用いた時，再起動と同時に問合せ実行開始した場合で，問合せ実行待ち時間を 57% 短縮した．早期ログ適用技法による，ログ適用にかかる時間へのオーバーヘッド

は高々 11% だった．

早期ログ適用による問合せ実行待ち時間の短縮効果は，単なる整列ログ適用による短縮効果を上回り，両者の組合せによりさらなる短縮効果が得られた．以上のように，早期ログ適用技法によって他では得られない問合せ実行待ち時間の短縮効果が得られることが確認できた．

6. まとめと今後の課題

本論文は，早期ログ適用によってログ適用中に発行された問合せ処理の実行待ち時間が改善する効果を，実験により明らかにした．

一般にデータベース管理システムは，ライトアヘッドロギングとチェックポイントを採用しており，障害等により再起動されると，データベースを格納するボリュームならびにログのディレクトリ情報等を探索し，最新のチェックポイントを同定し，当該チェックポイント以降に記録されたログをデータベースに適用し，データベースを回復し，一貫性のある状態となってから問合せ処理を受け付け，これの処理を実行する．よって，データベース管理システムに障害等が生じ再起動される場合，データベースを回復するまでの間，データベース管理システムは新たな問合せを受け付けて処理することができない．これはデータベース管理システムの可用性を低下させる 1 つの要因となっている．

早期ログ適用は，データベース管理システムが障害等により再起動された際に，ログの適用によるデータベースの回復を待つことなく，先行的に問合せ処理を実行可能とする技法であり，問合せがデータベースからページを取得する際には，当該ページが既に回復済みであって既にコミットされたトランザクションによる全ての更新が適用されているかどうかを判定し，そうであれば当該ページをそのまま利用し，そうでない場合には当該ページに未適用のログを取得することによりコミットされたトランザクションによる全ての更新が適用されている状態に回復させてから当該ページを利用する．

著者は，早期ログ適用機能を備えたデータベースエンジンを試作し，複数の更新偏りパターンの更新トランザクションによって生成された一連のログを対象として，当該ログの適用中に複数のデータアクセス偏りパターンを有する問合せを発行し，それぞれの組合せで，ログ適用ならびに問合せ処理に要する実行時間や実行中に各種資源利用量を計測し，問合せ処理の実行待ち時間の変化を評価した．この結果，早期ログ適用技法を用いることにより，ログ適用中に発行された問合せ処理の実行待ち時間が改善することを確認した．

上記の実験では比較的簡単なデータセットを用いて評価を行ったが，今後は TPC-C [5] をはじめとする標準的なベンチマークや現実のデータセットならびに負荷を用いてその有効性を確認する必要がある．また，オープンソースのデータベース管理システム等へ実装することにより，実装上の工夫や周辺ソフトウェアとの親和性等を丁寧に検証していく必要がある．

文　献

[1] R. Elmasri and S. Navathe, Fundamentals of database systems, Addison-Wesley Publishing Company, 2010.

[2] J. Gray and A. Reuter, Transaction processing: concepts and techniques, Elsevier, 1992.

[3] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz, “Aries: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging,” ACM Transactions on Database Systems (TODS), vol.17, no.1, pp.94–162, 1992.

[4] 塚井知之，加藤比呂武，“Oracle の高可用性技術とミッションクリティカル・システムへの適用，” 電子情報通信学会技術研究報告. DE, データ工学，vol.107，no.254，pp.63–68，2007.

[5] T.P. Council, http://www.tpc.org/.

[6] D. Shasha and P. Bonnet, Database tuning: principles, experiments, and troubleshooting techniques, Elsevier, 2002.

[7] A. Jhingran and P. Khedkar, “Analysis of recovery in a database system using a write-ahead log protocol,” Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 2-5, 1992., pp.175–184, 1992. http://doi.acm.org/10.1145/130283.130313

[8] H.F. Korth, E. Levy, and A. Silberschatz, A formal approach to recovery by compensating transactions, University of Texas at Austin, Department of Computer Sciences, 1990.

[9] M.J. Franklin, M.J. Zwilling, C.K. Tan, M.J. Carey, and D.J. DeWitt, “Crash recovery in client-server EXODUS,” Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 2-5, 1992., pp.165–174, 1992. http://doi.acm.org/10.1145/130283.130312

[10] V. Kumar and M. Hsu, Recovery mechanisms in database systems, Prentice Hall PTR, 1997.

[11] H.-T. Chou and D.J. DeWitt, “An evaluation of buffer management strategies for relational database systems,” Algorithmica, vol.1, no.1-4, pp.311–336, 1986.

[12] J. Pérez, et al., “Least likely to use: a new page replacement strategy for improving database management system response time,” International Computer Science Symposium in RussiaSpringer, pp.514–523 2006.

[13] J.S. Vitter, “Faster methods for random sampling,” Communications of the ACM, vol.27, no.7, pp.703–718, 1984.