効率的なWebDBサービス実現のための XML形式DBキャッシュ

小林真也, 川越恭二
立命館大学 理工学部

1. はじめに

最近のインターネット利用人口の増大や利用可能回線の拡大により多くの人が大容量のデータにアクセスすることが可能になった．しかし，コンテンツを提供するサーバ側では回線拡大に伴いデータ処理量の増加により負荷の増加という問題が発生している．膨大なデータアクセスがサーバのレスポンスを遅くする原因となっている．本稿では，サーバの負荷を軽減するためのサーバ側での新しいキャッシュ手法Cache-DBキャッシュを提案する．

2. 従来のキャッシュ技術

従来，多くの種類のキャッシュ手法が存在する[1]．クライアントサイドにプロキシサーバを配置し，アクセスされたオブジェクトをプロキシサーバに保存するProxy Cache や，サーバとクライアントの間にルーターを挟み，動的にキャッシュコントロールを行うTransparent Cache. 同じく動的にデータキャッシュサーバを選択するPush Cache. 履歴を調べ，利用頻度の高いオブジェクトをキャッシュとして保存し，常に最新状態として更新を行うActive Cacheなどがある．いずれもの手法でも効率的なキャッシングが行われる反面，クライアント側での設定が必要であったり，コスト面で非常に高くなる問題も存在する．また，サーバの状況によっては逆にトラフィックが増加することもある．主にクライアント側でのキャッシングではキャッシュ対象が非常に狭い範囲に絞られる可能性がある．そこで，サーバ側に近い位置でのキャッシュにより，より広範囲でのキャッシングが可能である．そこで，以降では，サーバ側でのキャッシングを対象とし新しいキャッシュ手法を提案する．

3. システムの概要

3.1. 基本的な考え方

一般的なキャッシュシステムではデータ更新時に既存のキャッシュを破棄する構造のものが多い．しかし，この方式では更新時ごとに破棄とキャッシュミスによるキャッシュの再作成が行われ，さらに更新に関する命令が多いために効率がよくない．提案する手法ではテーブル更新時にキャッシュを破棄せずに，一定の割合でキャッシュの更新を行うことにより，ヒット率の向上を図る．さらに，キャッシュの内部構造をデータベースと同様のものとしデータベースへ操作と同等の操作をキャッシュに対しても行うことができるものとする． サーバ内あるいはサーバと直結しているデータベース内のテーブルをキャッシュとして別に保存し，保存形式にはXMLを用いる．

3.2. 提案するキャッシュ法

先に述べた考え方に基づき以下に提案するキャッシュ方法Cache-DBキャッシュを説明する．

まず，ユーザはサーバ（通常Webサーバあるいはアプリケーションサーバ）に問い合わせを行う．サーバは，問い合わせに関係するテーブルがキャッシュ内に存在しているか，またそれが最新のデータであるかを調べる．該当テーブルが存在し，かつデータが最新であれば，キャッシュに対してクエリを発行し，該当データを取得する．なお，データ更新命令の場合には，データベースへの更新を通常通りおこなう．さらに，キャッシュではデータベースにアクセスせずに，同時にクエリを解析してツリー構造を最新状況に更新する．

図１に本方式の概要図を，図２にキャッシュのアルゴリズムを示す．

3.3. キャッシュ手法の実現

本方式におけるキャッシュはツリー構造であるXML形式で記述するが，このツリー構造の操作を容易にするため，開発言語としてXI（ザイ）[2]を用いた. XIは，

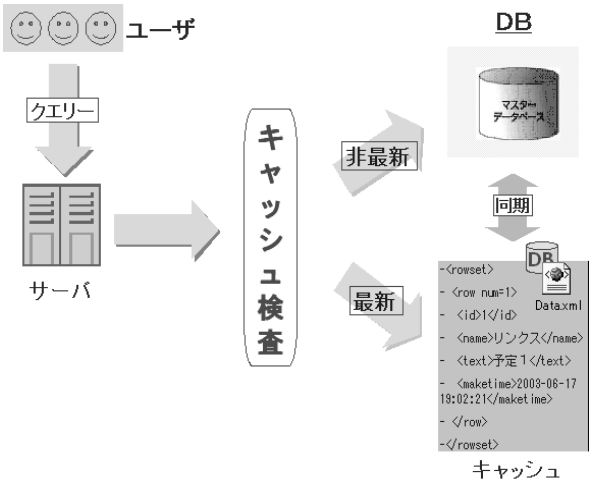

図１：キャッシュ概要図

XML Based DB Cache for Efficient WebDB Service Realization.
Shinya Kobayashi and Kyoji Kawagoe , College of Science and Engineering., Ritsumeikan University

```
If ( 選択系のクエリ ) {
    If (キャッシュが存在し,最新) {
        ・クエリを XML 対応形式に変換
        ・キャッシュからデータを取得
    } else {
        ・DB から最新データを取得しキャッシュを更新
        ・キャッシュが一定の容量を超えていたら
          LRU に基づき該当キャッシュを削除
        ・index(キャッシュ情報)を更新
    }
} elseif ( 更新系のクエリ ) {
    ・DB にアクセスしてデータを更新
    If (一定の確率でのキャッシュを更新が確定) {
        ・クエリを XML 対応形式に変換
        ・キャッシュを更新
        ・キャッシュが一定の容量を超えていたら
          LRU に基づき該当キャッシュを削除
    }
    ・index(キャッシュ情報)を更新
} else {
    ・DB に直接実行
}
```

図2：キャッシュアルゴリズム

Web アプリケーション開発に特化したプログラミング言語で，XMLとの親和性が非常に高い．

本方式では，データベースにアクセスされたSQL言語のクエリを分解し，XIのツリー制御の構文に変換する．特に，SQL命令での対象となるデータを特定する構文（Where句など）は，XIにおけるツリー内のノードの位置を指定する記述形式（**XPath**）に変換する．

4．評価

4．1．実験環境

キャッシュヒット率を検証するため，(1)更新が発生するたびにキャッシュを破棄する方式(Check & destroy)，(2)提案方式であるキャッシュを連続的に更新していく方式（Continuous update）について，両者を比較する実験を行った．ここで，(2)のキャッシュシステムでは，キャッシュ更新率をデフォルトとして 50%と設定した．これは，データベースが更新される際に 50%の割合でキャッシュも同時に更新することを意味する．

なお，実験に使用する DBMS は HSQL である．HSQL はピュア JAVA の単純な仕組みのデータベースのため，一般的な DBMS より動作が速いという特徴がある．

4．2．実験データ

実験対象として,単純なデータを含む4種類のテーブルを作成した．ランダムなデータベースへのアクセス命令（SQL によるクエリ）を 1000 回連続で実行し，キャッシュヒットによるキャッシュへのアクセス回数を測定した．データベースへのランダムな命令では，一定の割合でデータベースの更新を行うような命令を含んでいる．

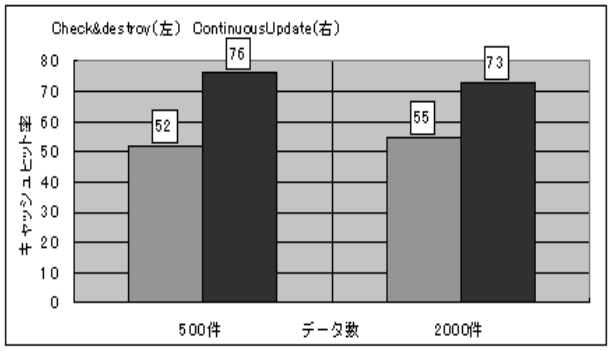

図3：DB のデータ数の変化によるキャッシュヒット率

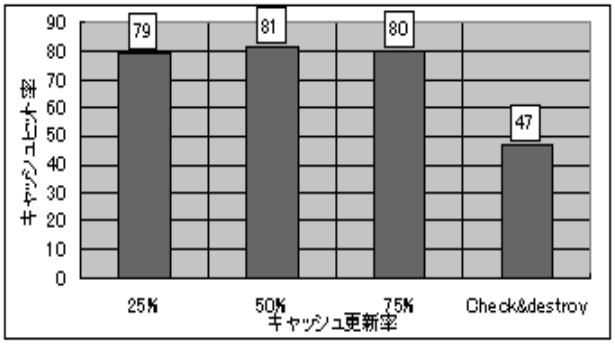

図4：キャッシュ更新率の変化によるキャッシュヒット率

4.3．評価

図3,図4に実験結果を示す．

図3はキャッシュの更新率を変えずに，データの件数を変化させたときのヒット率である．双方の結果の左部のグラフ棒が従来のキャッシュ破棄方式，右部が提案方式の結果である．

図4はキャッシュの更新率を変化させたときの結果である．最も右に示す従来手法ではデータ数を1000件とした結果である．

いずれの図でも，提案方式のキャッシュヒット率は，従来の方式と比較して高いことがわかる．

5．おわりに

本稿では，サーバ側での新しい DB キャッシュ方法 Cache-DB キャッシュ方法を提案した．試作システムによりキャッシュヒット率が大幅に向上し，より効率的なアクセスが可能となった．今後は、Cache-DB キャッシュ方法の効率的実現および複雑な問い合わせへの対応等を行う予定である。

参考文献

[1] G.Barish, K.Obraczka:
"World Wide Web Caching: Trend and Techniques"
IEEE Comm.Magazine May 2000

[2]横浜ベイキット:
http://www.baykit.org/index.