日本ソフトウェア科学会第 28 回大会 (2011 年度) 講演論文集

Cassandra と非破壊的構造を用いた CMS のスケーラビリティ検証環境の構築

玉城 将士　　河野 真治

本研究では，スケーラビリティのある CMS を開発するために，PC クラスタを利用したスケーラビリティの検証環境を構築し，ロックフリーな木構造である非破壊的木構造・多段キャッシュ・Cassandra を用いた設計と実装を行って来た．今回は，実装したシステムのスケーラビリティを検証するため，構築したスケーラビリティの検証環境を用いてベンチマークを取り，スケーラビリティがあるか確認するために，環境構築を行った．また，非破壊的木構造をバランス木に応用し，バランス木の性能である O(log N) を保ちつつ並列に読み・書き込みが可能である辞書アルゴリズムの提案をする．

1　はじめに

Cassandra は複数のサーバーで動作を想定した分散データベースである．本研究は，Cassandra の検証と非破壊的木構造を用いたスケーラビリティのある CMS の設計と開発を行った．非破壊的木構造を用いた CMS のとは，木構造で表すことの出来るコンテンツを編集する際に，編集元の木構造を破壊することなく編集するアルゴリズムである．これを利用して Cassandra 上に非破壊的木構造を構築し CMS を実装することができた．

本研究では，開発した CMS のスケーラビリティの検証を行うため，仮想環境を用いた検証環境の構築と管理ソフトウェアを開発した．

2　分散データベース Cassandra

Cassandra は，FaceBook が自社のために開発した分散 Key-Value ストアデータベースであり，Dynamo?[] と BigTable?[] を合わせた特徴を持っている．2008 年にオープンソースとして公開され, 2009 年に Apache Incubator のプロジェクトとなった．2010 年には Apache のトップレベルプロジェクトとなり, 現在でも頻繁にバージョンアップが行われている．

3　非破壊的木構造

非破壊的木構造とは，木構造を編集する際に編集元の木構造を破壊することなく，新しく木構造を構築する．新しい木構造のルートノードを置き換えることにより編集する方法である．
非破壊的に変更することで，編集元の破壊することなく編集することが出来るため，木構造の整合性を保ちつつ変更することが可能になる．

3.1　木構造の破壊的変更

従来の破壊的木構造は，存在する木構造を書き換えて編集する．以下の様な操作を行う．

図 1 の操作では，ノード F の内容をノード G に書き換える操作を行った．破壊的変更では，単純に編集したいノードを書き換えることにより行われる．この操作では，編集時に木を参照している処理がある場合，参照されている木構造を破壊するため，参照を開始した自転での木構造の整合性が破壊されるという問題が起きる．

Constructing Scalability Evaluation Environment for CMS using Monotonic-Tree Operation and Cassandra

Shoshi TAMAKI, Shinji KONO, 琉球大学理工学研究科情報工学専攻, Dept. of Information Engineering, Ryukyu University.

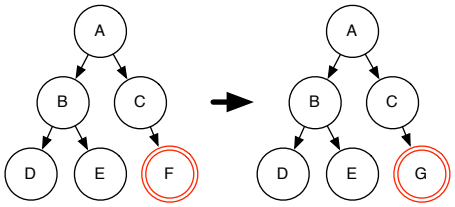

図 1　木構造の破壊的変更例

この問題を解決するためには，木構造の操作に排他

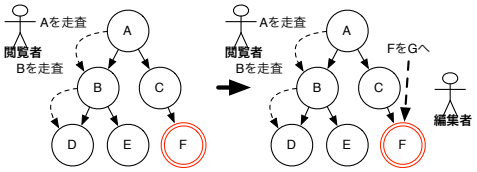

図 2　破壊的変更の問題点

制御を取り入れてロックする必要がある．しかし，その方法ではスケーラビリティを確保できるとは考えられないため，非破壊的な変更を方法を用いて木構造を編集する．

3.2　木構造の非破壊的変更

木構造の非破壊的な変更は，編集元の木構造を破壊せずに編集を行う，編集の様子を図 3 に示す．図 3 では図 1 と同様にノード F の内容をノード G に書き換える処理を行っている．

この方法での編集は以下の手順を用いて行われる．

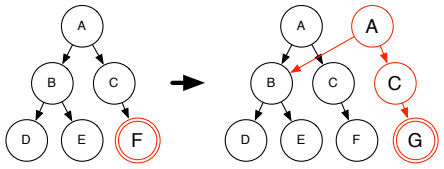

図 3　木構造の非破壊的変更例

1. ルートノードであるノード A から編集対象であるノード F までのパスをコピーする．(ノード A, C, F をコピーする)
2. パスに含まれないノードは編集元のノードと共有する．(コピーノード A からノード B へリンクを作成する)
3. 編集対象であるノード F は編集せず，コピーしたノード F をノード G へと編集する．
4. 木構造のルートノードをノード A からコピーしたノード A へと置き換える．

この手順では，元の木構造は破壊されることは無い，そのため木の閲覧者が存在していても閲覧している木構造の整合性が破壊されることはない．よって，並列に読み書きを行うことが出来る．

3.3　応用例：非破壊的木構造を用いた二分木辞書

この方法の応用例として非破壊的木構造とバランス木を用いた二分木辞書を考えることが出来る．二分木辞書とは二分探索を用いた O(lg n) を保証する辞書アルゴリズムである．二分木辞書では，バランスのとれた木構造を維持するためにバランス木のアルゴリズムを利用する．これらのアルゴリズムと非破壊的木構造を組み合わせることにより，並列に読み書きを行うことが出来る辞書を作成することが出来る．また，この辞書アルゴリズムの利点として辞書全体のコピーにかかる計算量が O(1) で済むことも利点の一つである．

3.3.1　AVL-Tree を用いた非破壊的二分木辞書

実装例として，AVL-Tree を用いた非破壊的二分木辞書を紹介する．この辞書は読み書きが O(lg n) かつ辞書の複製のコストが O(1) で有ることを保証する．

1. 辞書の読み込み

   非破壊二分木辞書の読み込みは通常の二分木と同様で，ルートノードよりキーの大小関係を比較し値を検索する．そのため，省略する．キーを検索する際に，二分木辞書で使われている木構造は破壊されることがないため，並列に行うことが出来き，スレッドセーフである．
2. 辞書の書き込み

   非破壊辞書の書き込みは以下の手順で行われる．

   (a)　二分木探索より，書きこむ場所を特定する．この時，同時に通過したノードのコピー

を行う．

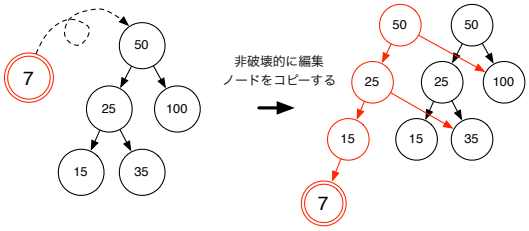

図 4　手順 1:ノードのコピーと書き込み

この例では，木構造に新しくノード 7 を追加する．そのため，編集元の二分木より $50, 25, 15$ のノードをコピーする．他の影響のない $100, 35$ は共有する．

(b) ローカルにコピーしたノードを編集し，書き込みむ

次に，コピーした木構造を編集し書き込みを行う．図 4 の例ではノード 15 の右部分に新しくノード 7 を追加する．

(c) コピーし編集したノードよりルートノードまでを走査し，木の回転が必要な場合は回転させる．

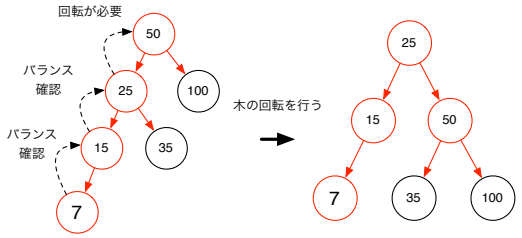

図 5　手順 2:コピーした木構造のバランス

新しく構築した二分木のバランスさせるために木構造を追加したノードよりルートノードまで辿りバランスを確認する．図 6 ではノードを $7 - 15 - 25 - 50$ とルートへとバランスを確認し，回転が必要であるノード 50 の位置で木の回転を行う．

(d) CAS を用いて，ルートノードへの参照を入れ替える．最後に，二分木辞書がルートノードとして保持している編集元の木構造を，新しい木構造へと置き換える．この時 CAS を使用することによりアトミックに置き換える．他のスレッドがこの木構造を編集し置き換えていた場合，この処理は失敗する．その場合，再度，非破壊的に編集を行う．

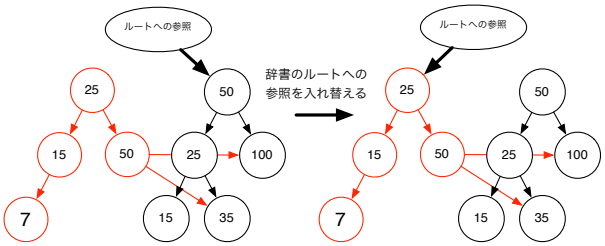

図 6　手順 3:非破壊的に編集した木構造の適用

3. 辞書のコピー

非破壊辞書のコピーは，単にルートノードを共有するだけで行うことが出来る．木構造は破壊されないため，元の木構造は不変である．共有した木構造を元にローカル新しい木構造を作成していくため，問題は起きない．

よって，この場合の計算量は定数であり O(1) である．

この二分木辞書は主に，辞書をコピーするときに効果を発揮する．

4　非破壊的木構造を用いた CMS

本研究では，非破壊木構造を用いてスケーラビリティのある CMS の設計と実装を行った．本システムではコンテンツを木構造で表現する．Cassandra 上に木構造を構築し，それを非破壊的に編集する．図 7 に概略図を示す．

本システムでは，Cassandra 上に木構造を構築するサーバー (API Server) を設ける，サーバーの提供する API を用いてコンテンツを非破壊的に操作することができる．WebServer は API Server を利用してコ

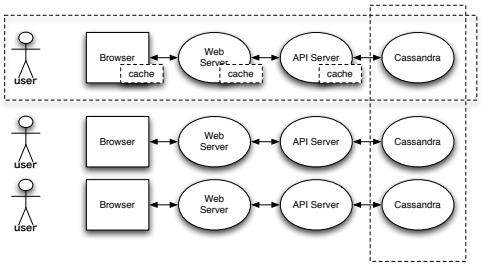

図 7　システムのアーキテクチャ

ンテンツを操作しコンテンツの配置を記述したレイアウトを用いてレンダリングを行い，木構造を編集する際には専用のエディタを提供する．(図 8) また，各段階 (API Server , WebServer , Browser) で木構造のキャッシュを保持し，必要になったときのみキャッシュを同期・マージさせる．(図**??**) こうすることでスケーラビリティを確保することが出来ると考えられる．

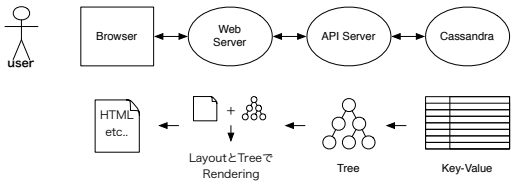

図 8　木構造のレンダリングと編集

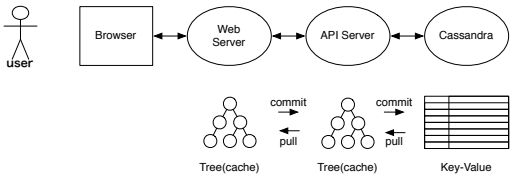

図 9　多段キャッシュとマージ処理

5　検証環境の構築

本検証では，前回行った PC クラスタによるスケーラビリティの検証環境とは異なり，仮想環境上に検証環境を構築する．仮想環境のホストとして利用するサーバーを表 1 に示す.

表 1　検証環境に用いたサーバー

サーバー名	CPU	メモリ	仮想化
server01	Xeon x2	139GB	KVM

5.1　仮想環境

5.2　仮想化管理ツールの実装

5.2.1　libvirt

libvirt とは

5.2.2　webvirt

webvirt とは，本研究室で開発した仮想環境の管理ツールである．cakephp+libvirt

6　まとめ