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データベースある. 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 [2] と BigTable [1] 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り, 現在でも頻繁にバージョンアップが行われている. 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.

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.

## 2 非破壊的木構造

非破壊的木構造は, 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, 編集元の木構造を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.
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,

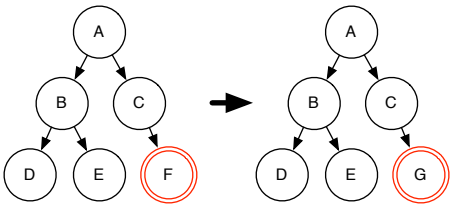


**図 1　木構造の破壊的変更例**

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, 参照を開始した時点での木構造の整合性が破壊されるという問題が起きる． この問

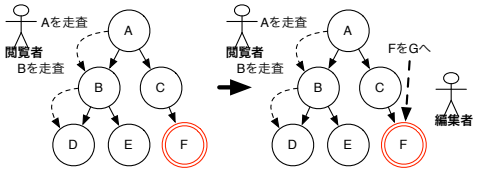


**図 2　破壊的変更の問題点**

題を解決するためには, 木構造の操作に排他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, 非破壊的な変更を用いて木構造を編集する．

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．

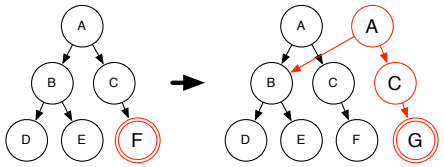


**図 3　木構造の非破壊的変更例**

この方法での編集は以下の手順を用いて行われる．

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く, 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　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．

### 3.1　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．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 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．

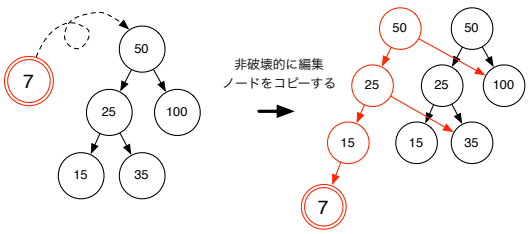


**図 4　手順 1:ノードのコピーと書き込み**

(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.

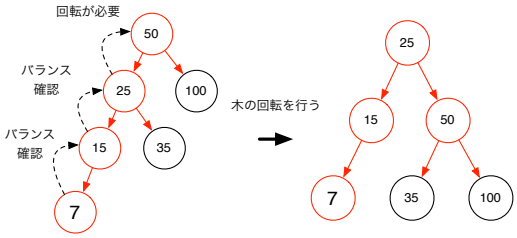


**図 5　手順 2:コピーした木構造のバランス**

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.

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.

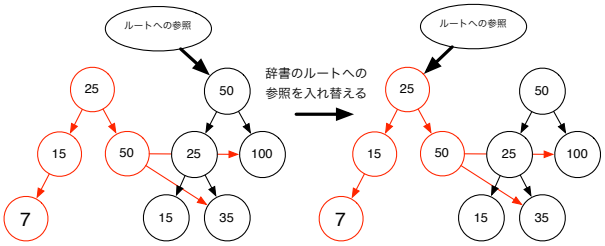


**図 6　手順 3:非破壊的に編集した木構造の適用**

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. 本研究で開発した非破壊的構造を用いた CMS では, 木構造を編集する際に, ノードの持つ属性を同時にコピーする. 本来は単純な辞書のコピーで合ったが, これを非破壊的二分木辞書を利用することで改善することが出来るのではないかと考えられる.

## 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. [4] 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

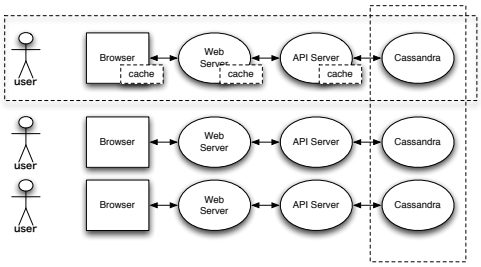


**図 7　システムのアーキテクチャ**

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ている. 各段階のキャッシュは, 親の木構造に対してコミット, マージ処理を行うことができ, 分散レポジトリと同様の機能を提供している. 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 9)

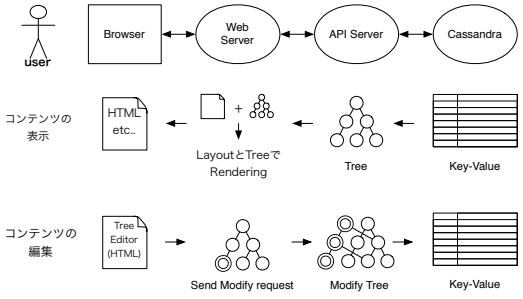


**図 8 木構造のレンダリングと編集**

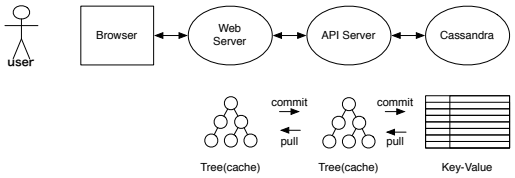


**図 9 多段キャッシュとマージ処理**

## 5 検証環境の構築

検証では, 新しく導入されたブレードサーバーによる仮想環境において検証環境を構築する. [3] 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.

### 5.1 仮想環境

仮想化とは１つの物理マシン上にて複数のオペレーティングシステムを動作させる技術である. 物理マシンにハイパーバイザと呼ばれる物理マシンのリソースを仮想化するレイヤを追加する. ハイパーバイザは仮想マシンを作成し仮想化したリソースを仮想マシンに提供する. その上でオペレーティングシステムを動作させることで, 複数のオペレーティングシステムの稼働を可能にする. ハイパーバイザは複数ありVMWare,KVM,Xen,Hyper-V などがあげられる. 本検証では, KVM と VMWare を用いた検証を行う予定である.

**表 1 検証環境に用意したサーバー**

| サーバー名 | CPU | メモリ | 仮想化 |
|---|---|---|---|
| server01 | Xeon X5650 x2 | 130GB | VMware ESX |
| server02 | Xeon X5650 x2 | 130GB | VMware ESX |
| server03 | Xeon X5650 x2 | 130GB | VMware ESX |
| server04 | Xeon X5650 x2 | 130GB | KVM |

### 5.2 仮想環境を用いた検証方法の検討

仮想環境を用いた検証方法は基本的に前回 [3] の PC クラスタを用いたスケーラビリティ検証と同様の方法を採用する. (図 10) つまり, 並列アクセス用のクライアントクラスタとサーバー用のクラスタを用意する. 今回は, server01-03 をクライアントクラスタ用の仮想環境に, server04 をサーバークラスタ用の仮想環境として用いる. 今回の検証では, PC クラスタを用いるのではなく仮想環境利用する. 仮想環境において複数の仮想マシンで物理マシンのリソースを共有することになるため, 仮想マシン同士が物理マシンのリソースを奪い合う可能性が出てくる. リソースの奪い合い

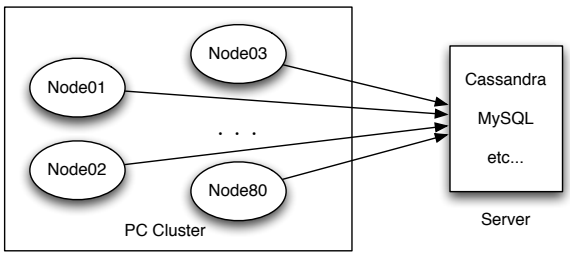


**図 10 PC クラスタを用いたベンチマーク方法**

による, 仮想マシンの性能低下を防ぐため, 仮想マシンのリソースを予約・制限する必要がある. (図 11) 仮想マシンのリソースを予約・制限すると, 制限以上の性能は出なくなる. それを応用し, 物理マシンのリソースの範囲内で同様の仮想マシンを構築することにより台数効果も検証できるのではないかと思われる.

以上のように仮想環境を構築し仮想環境において検証を行う. 仮想環境の構築において, 複数の仮想マシンを操作することが必要となる. 通常, 物理マシンのコンソールより個々の仮想マシンを操作する作業は効率が悪いため, 管理ツールの開発を行う.

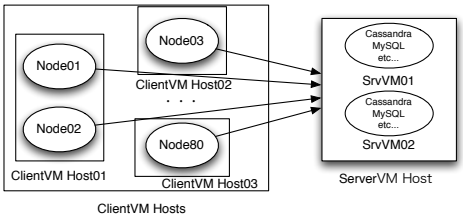


**図 11　仮想環境を用いたクラスタ環境の構築**

### 5.3　仮想化管理ツールの実装

仮想環境で複数の仮想マシンを操作する場合, 仮想マシン個々の設定を物理マシンのコンソールより操作するのは大変困難な作業であり, 仮想化管理ツールの利用が必須であると考えられる. そこで, 本研究では初めに仮想環境を管理するツールを開発し, 検証環境の構築に利用する.

#### 5.3.1　libvirt

libvirt とは, 複数ある仮想環境においてノードをリモートより共通で十分に安全な安定した管理方法を提供するライブラリである. この場合のノードは 1 台の物理的なマシンであり, ドメインは仮想マシンを指す. 様々な言語とハイパーバイザ, ユーザーの認証方法に対応している.

1. ハイパーバイザ
   KVM,Xen,VirtualBox,VMWare,etc..
2. プログラミング言語
   C,Python,C#,Java,Perl,Ruby,PHP,etc..

libvirt を用いた仮想化管理ツールは複数存在している, 以下にその一例を示す. `libvirt` を用いた管理ツールは複数存在するが, インストールが複雑であり, 必要のない機能を実装してることが多い. そこで, 導入が用意であり, かつ十分な機能を提供するウェブ上の管理ツールを実装する.

**表 2　libvirt を用いた管理ツール**

| | |
|---|---|
| virsh | CUI |
| virt-manager | GUI |
| oVirt,AbiCloud,etc... | Web |

#### 5.3.2　webvirt

`webvirt` とは, 本研究室で開発した仮想環境のウェブ管理ツールである. ウェブアプリケーションフレームワークである CakePHP と libvirt-php を用いて開発した. webvirt は, シンプルで十分なシングルノード上でのウェブ仮想化管理ツールを目指して開発した. インストールに必要なのは PHP が動作可能なウェブサーバー・PHP・libvirt-php のみであり, 主な機能として, 以下の機能を提供する.

- VNCViewer(TightVNCViewer2)
- 仮想マシンのシャットダウン
- 仮想マシンの起動
- 仮想マシンの定義・変更
- ストレージプールの管理
- ネットワークの管理

また, シングルノードのみを管理する目的で開発されているため, ライブマイグレーションなどの機能は実装していない.

## 6　まとめ

本研究では, 開発した非破壊的構造を用いた CMS のスケーラビリティ検証を行うため, 仮想環境における検証環境の構築方法について検討した. 基本的な方法は前回 [3] と同様に行い, クライアントクラスタとサーバークラスタの仮想環境を構築する必要がある. 仮想環境下では, 仮想マシンによる物理リソースの取り合いを防ぐため, 仮想マシンのリソースを予約・制限しつつ構築する必要があることが分かった. 仮想

マシンは複数存在するため管理が困難だと考えられ, ウェブ上の管理用ツールを開発した．また, 非破壊的構造の応用例として並列に読み書きを行うことの出来るバランス木を用いた二分木辞書の実装例を示した. 次回は, 構築した仮想環境によるスケーラビリティの検証を行う予定である.

## 参考文献

[1] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber: Bigtable : A Distributed Storege System for Structured Data

[2] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati , Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall , Werner Vogels: Dynamo: Amazon's Highly Avaliable Key-value Store , SOSP (2007)

[3] Shoshi TAMAKI,Shinji KONO:Cassandra を用いた CMS の PC クラスタを用いたスケーラビリティの検証, ソフトウェア科学会 (2010)

[4] Shoshi TAMAKI,Shinji KONO,Yu TANINARI: Cassandra を使ったスケーラビリティのある CMS の設計, 情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS)