Vol.2015-ARC-215 No.20
Vol.2015-OS-133 No.20
2015/5/27

非破壊的木構造データベース Jungle とその評価

金川竜己[†1]　河野真治[†2]

我々が扱っている知識は、書籍や組織情報など木構造であることが多い。しかし RDB は木構造をそのまま格納するのには向いていない。そこで当研究では、非破壊的木構造データベース Jungle を提案している。Jungle は、属性名と属性値の組と持つ Node を持ち、Node は子ノードのリストを持ち、木構造を構成する。名前を持つ複数の木の集合が JungleDB となる。Jungle のトランザクションは名前と木の結合をアトミックに書き換えることで実現される。この時に元の木を変更しない特徴がある。これにより、書き込みと読み込みの並行実行を確保している。Jungle 上に業務用許認可システム maTrix を構築し、性能評価を行った

Evaluetion of Non Destractive Tree structured DataBase Jungle

Tatsuki KANAGAWA[†1] and Shinji KONO [†2]

The knowledge we use in many are usually represented in tree structures, such as books or structure of organization. The relational database has flat table structure, which is not suitable store present the tree structures. So we introduce jungle database, which has non destructive tree structure.

Jungle database has nodes which have dictionary of at true name and its value. A node has children it has. A set of named tree is a Jungle database. transaction of jungle is atomic replace of name binding of the tree. Previous tree is preserved in the transaction. Since transactions preserved old tree, read and write runs safely in parallel. make an application of authorization of resource in an organization. Compare of Jungle and Mongo DBs is also Performance.

1. 研究背景と目的

我々が扱っている知識は木構造であることが多い。例えば書籍や組織情報などである。木構造のデータをそのままデータベースに格納することで直接的な操作や効率的なサービスが可能になると考えられる。しかし RDB は木構造をそのまま格納するのには向いていない。そのため、それらの構造を格納するのに特化した NoSQL というデータベースがある。例えば、Cassandra や mongoDB などである。当研究室では、これらの問題を解決した煩雑なデータ設計が必要のないデータベースを目指して非破壊的木構造データベース Jungle を開発している。非破壊的木構造とは、データの編集の際に一度保存したデータを変更せず、新しく木構造のデータを作成してデータの編集を行うことである。そのため、読み込み中にデータが変更されないことが保証されているため、書き込みと読み込みを同時に行える。Jungle は、これまでの開発によって木構造を格納する機能を持っている。

本研究では、Jungle 上に組織に許認可管理アプリケーション maTrix を実装し、データベースの表現力、機能の十分性、実用的な性能があるか、実証実験を行う。Jungle の評価は、業務アプリケーション maTrix を Jungle と mongoDB 上に実装し、読み込みの速度比較と、read と write と read を並列の動作させた場合の 1 秒間の readCount 数の比較の 2 つを行った。mongoDB は、データを Json に似た Bson 形式で扱っている NoSQL データベースである。Jungle と似たデータ構造を持っているため比較対象には mongoDB を選択した。

2. 許認可管理アプリケーション maTrix

maTrix は、人、役職、役割、権限、組織等の木構造のデータと、ポリシーファイルを持つ。maTrix の組織構造は、データ同士が参照を行うことで表現される。また、組織構造は版管理されている。

ポリシーファイルは、データに対するアクセス要求が許可されるか否認されるかを判断するためのルールを誰が (Target)、何を (Redource)、どうできるか

†1 琉球大学理工学研究科情報工学専攻
Interdisciplinary Infomation Engineering, Graduate School of Engineering and Science, University of the Ryukyus.
†2 琉球大学工学部情報工学科
Infomation Engineering, University of the Ryukyus.

Vol.2015-ARC-215 No.20
Vol.2015-OS-133 No.20
2015/5/27

(Action) の 3 つの要素で記述されている。maTrix はアクセス要求に応じた、ポリシーファイルを参照することで許認可の判断を行う。ポリシーファイルは、組織構造中の人や役職を id を用いて参照している。つまり、ポリシーファイルを用いて許認可の判断を下すためには、その人がどの組織に所属して、その役割がどの権限を持っているかを返す検索が必要となる。

3. 非破壊的木構造データベース Jungle

Jungle は複数の木の集合からなり、木は複数のノードの集合で出来ている。ノードは自身の子ノードの List と、属性名と属性値の組を持つ。Jungle は、非破壊的木構造であるためデータの編集は、一度生成した木を上書きせず、データの編集はルートから編集を行うノードまでコピーを行い新しく木構造を構築することで行う。そのため、読み込みと書き込みを同時に行うことができる。他にも、本研究室で開発を行っている分散フレームワーク Alice を用いて作られた分散木構造データベース Jungle もある。

4. Jungle 上での maTrix のデータ構造の表現

maTrix の人、組織、役割、権限等のデータ構造は木構造なので、Jungle の木構造にそのままマッピングできる。実際の maTrix のデータ構造の一部を格納した JungleTree(図 1) を以下に記す。

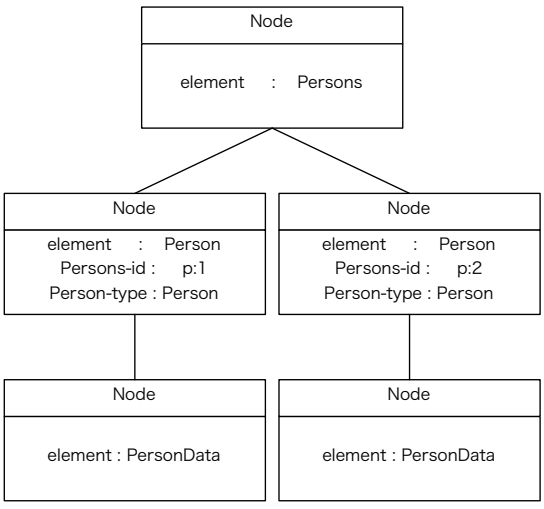

図 1 Jungle 上での人物 Tree の表現例 (1)

Jungle は、TreeNode にデータを格納する際、String 型の属性名と ByteBuffer 型の属性値の組み合わせで保持しているため、1 つの属性名に対して複数の属性値を持つことは出来ない。そのため、表 1 の様に、1 つの要素に複数の値がある場合はデータを格納できない。しかし、表 2 の様に、データを分割し、2 つの Node に分けて格納することで、Jungle に格納できるようになる。

表 1 Jungle 上で表現できないデータ例

<ids>r:10 r:34</ids>

表 2 Jungle に対応したデータ例

<id>r:10</id>
<id>r:34</id>

Jungle 上での maTrix の組織構造の表現は木に対する id の検索を用いて表現すれば良い。

また、maTrix は自身のデータを XML 形式で書き出すことが可能である。書きだしたデータを Jungle に格納するために JungleXMLReader の実装も行った。

5. 検 索 API

Jungle 上での maTrix の組織構造の表現は Id を用いた木の検索を用いて表現するため、Jungle に属性名:属性値の組で検索を行う API の実装を行った、

Jungle の Tree に対する検索は、lambda 式を用いて find 関数を実装した。lambda 式を使用することで、匿名クラスを使用した場合より簡潔にコードを記述できるようになった。以下に find 関数の定義を記す。

```
public Iterator<TreeNode> find(Query query,
            String key, String searchValue);
```

find 関数は引数に Query、String key、String value の 3 つの引数を取り、条件に一致した Node の Iterator を返す。第 1 引数には、探索の条件を記述する関数 boolean comdition(TreeNode) を定義した InterfaceQuery を。第 2、第 3 引数の、String key、String value は Index の取得を行うために使用する。find 関数の使用例を以下に記す。

find のサンプル

```
InterfaceTraverser personTraverser
= personTree.getTraverser(true);
Iterator<TreeNode> personIdpairIterator
= personTraverser.find((TreeNode node) -> {
    String personId =
    node.getAttributes().getString("Personid");
    if (personId == null)
      return false;
    if (personId.equals("p:4"))
      return true;
    return false;
}, "Personid", "p:4");
```

Vol.2015-ARC-215 No.20
Vol.2015-OS-133 No.20
2015/5/27

上記コードの Query 内での処理について解説する。

(1) 引数のノードから getAttributes() .getString("Personid") で属性名が Personid の属性値を取得する。
(2) 属性値が null だった場合、この Node には属性名が Personid の組のデータは存在しないので false を返し次の Node の評価を行う。
(3) 属性値が null でなかった場合、p:4 と一致するかどうかを調べ結果を返す。

6. Index の実装

Jungle のデータ検索は、木の全探索であるため計算量は O(n) である。しかし、Index を実装することで O(logN) で探索を行うことが可能になる。Jungle は、過去の木のデータを全て保持しており、アクセスすることが可能であるため全ての version で Index を保持する必要がある。よって、Index を破壊すること無く更新する必要があったため、非破壊的木構造の TreeMap を新しく自作し使用した。この TreeMap は、データの更新を行った際、前の版の TreeMap とデータを最大限共有した新しい TreeMap を作成する。これにより、複数の版全てに対する Index をサポートすることが可能になった。

以下に Index の型を示す

```
TreeMap<String key,TreeMap<String attribute,
        List<TreeNode> node> index> indexList
```

Jungle の Index は IndexList 内に保持されている。属性名で IndexList に検索を行うと、対応した Index が取得できる。その取得した Index に属性名で検索を行うとノードのリストが返ってくる。

他にも、渡したノードの親を返す ParentIndex の実装も行った。ParentIndex を使用することで、木構造の親を取得する検索も行えるようになった。

7. 新しい非破壊 TreeMap

Jungle 内で実装している Index や、Node の attribute を、FunctionalJava の TreeMap を用いて実装していた。しかし、測定を行った結果 FunctionalJava の TreeMap 部分がネックとなり性能が低下していたため、新しく非破壊的木構造の TreeMap の実装を行った。自作 TreeMap の構造は、データの検索、編集時の計算量から赤黒木を採用した。赤黒木とは、二分木の一種で、次のような特徴がある

- 各ノードは赤か黒の色をもつ。
- 根は必ず黒。
- 葉は全て黒。
- 赤のノードの子ノードは必ず黒。
- 全てのノードから子孫の葉までの経路に含まれる黒ノードの数は全て同じ。

上記の条件より、根から葉までの最長経路が最短経路の 2 倍以上にならないため、極めて高速にデータの検索が行える。

TreeMap の実装を行う際に、赤と黒のノードを StatePattern を用いて、可読性を向上させた。TreeMap を実装したことにより、Jungle がより高速に動作するようになった。

8. Benchmark

実験環境

- Mac OS X 10.10.2
- 2*2.66 GHz 6-Core Intel Xeon
- Memory 16GB 1333MHz DDR3
- java 1.8.0-45
- mongoDB 3.0.2
- javascript V8JavaScriptengine

図 2 は mongoDB と Jungle の速度比較のグラフである。比較は 10000 人分のデータに対するアクセスの処理時間で行い、mongoDB へのアクセスは js を用いて行った。

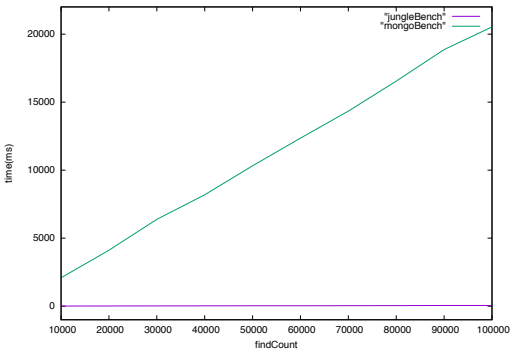

図 2　mongoDB との比較

Jungle は、mongodb の約 500 倍の性能が出ている。その理由として、mongoDB が mmap を用いてディスクのデータにアクセスしているのに対し、Jungle は全てのデータがメモリ上にあること。また、mongoDB は通信を介してアクセスされるが、Jungle は通信を介さないためだと考えられる。

図 3 は、Jungle の書き込みが、どれだけ読み込みに影響があるかを測定したグラフである。複数スレッドから Jungle に読み込みだけを行った場合と、1 スレッドだけ書き込みを行い、残りのスレッドで読み込み行った場合で測定を行った。

Jungle では書き込みと読み込みを同時に行っても性能低下はおこらなかったため、設計通りの性能が出たといえる。

Vol.2015-ARC-215 No.20
Vol.2015-OS-133 No.20
2015/5/27

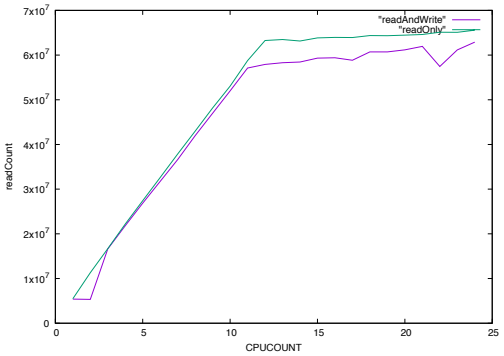

図 3　TransactionPerSecond

9. まとめと今後の課題

本研究では、Jungle 上に許認可管理アプリケーション maTrix の実装を行い、Jungle に実用データベースとしての表現力、機能の十分性、実用的な性能があるか実証実験を行った。maTrix は複数の木がお互いに Id を用いた参照を行い組織構造を表現していたが、Jungle では Id を用いた検索を行いそれを表現した。また、測定の結果 mongoDB より高速に動作したため、Jungle は、実用アプリケーションを実装できるほどの表現力、機能、性能があることを証明できた。

今回の研究では、maTrix が書きだした xml ファイルをそのまま Jungle に格納し性能測定等を行った。しかし、このデータ構造は最適化の余地があり、Jungle に向いたデータ構造にすることでさらなる性能の向上が見込むことができる。また、木のサイズが大きくなると変更がルートに集中するため、木を分割する必要があるが、分割した木同士の参照は Id を用いた間接的なものとなる。このように、Jungle は RDB と異なり格納するデータの自由度は大きい。なので、Jungle の設計手法を確立させる必要がある

参　考　文　献

1) 玉城将士，谷成　雄，河野真治：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) (2011).
2) 大城信康，　杉本優，河野真治，永山辰巳：Data Segment の分散データベースへの応用，日本ソフトウェア科学会第 30 回大会論文集 (2013).
3) : XACML References and Products, `http://docs.oasis-open.org/xacml/xacmlRefs.html`.
4) : The MongoDB 3.0 Manual, `http://docs.mongodb.org/manual/`.
5) : FunctionalJava, `https://github.com/functionaljava/functionaljava`.