【VLDB 2010 勉強会】

Session 10: Novel/Advanced Applications

担当：上田 高徳（早稲田大学）

Session 10: Novel/Advanced Applications 概要

（論文1）“Navigating in Complex Mashed-Up Applications”

- Daniel Deutch (Tel-Aviv University, Israel), Ohad Greenshpan (Tel-Aviv University & IBM Research Labs, Israel) and Tova Milo (Tel-Aviv University, Israel)

（論文2）“On Graph Query Optimization in Large Networks”

- Peixiang Zhao and Jiawei Han (University of Illinois at Urbana-Champaign, United States of America)

（論文3）“Dremel: Interactive Analysis of Web-Scale Datasets”

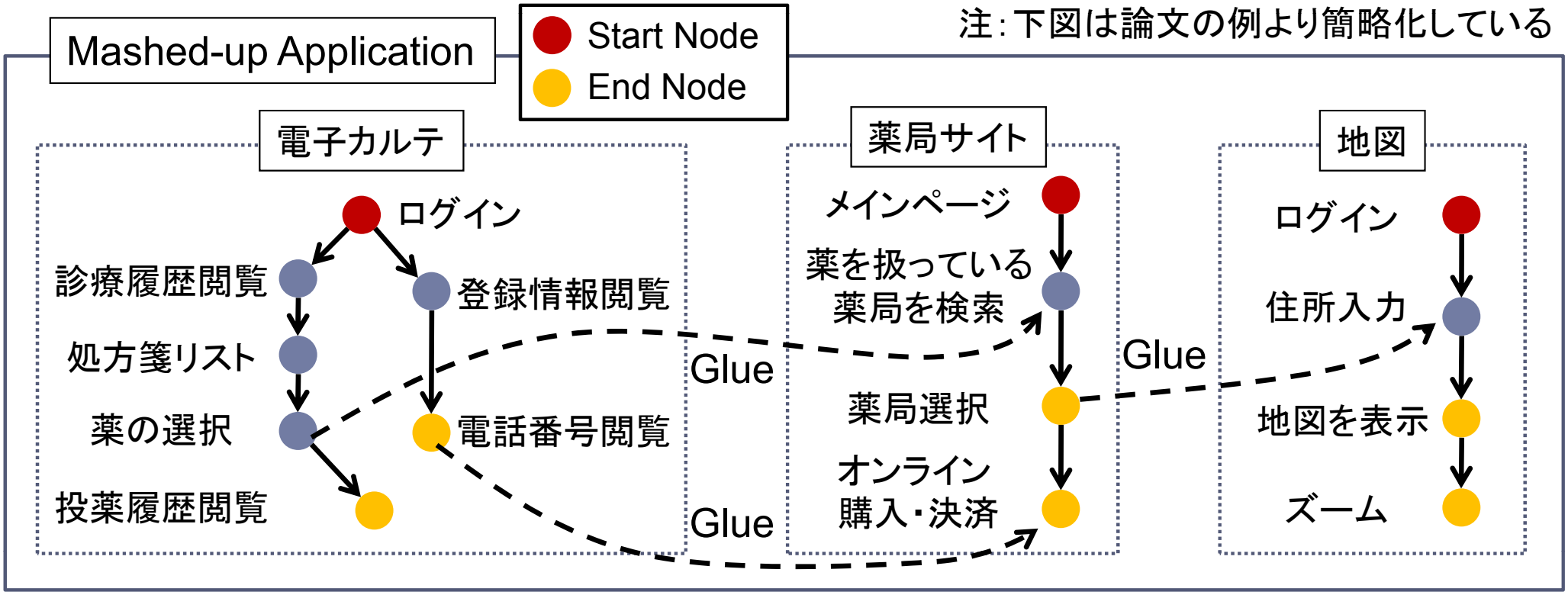
- Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis (Google, United States of America)

説明の都合上、ここでの番号は、VLDBのプログラム順とは異なります。

- 論文をより理解するためにあった方がよい知識
  - 論文1 と 2
    - 特になし？（問題定義から始まり、それを解く論文）
  - 論文3
    - MapReduce およびその特性（背景の理解に）
    - カラムストアの知識（手法の理解に）
    - 分散DBの実行モデル（実験の理解に）

（論文1）“Navigating in Complex Mashed-Up Applications”

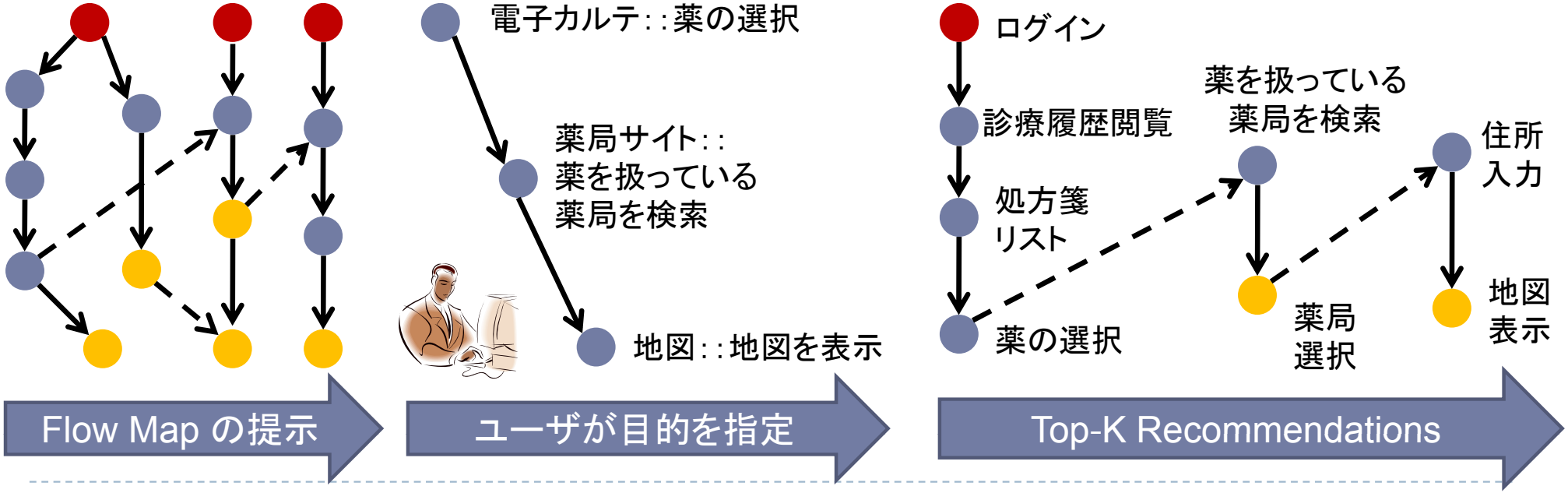
- （目標）複雑なマッシュアップアプリケーションにおけるユーザのナビゲーションを補助する
- 1つのApplication は機能（Mashlet）が有向グラフで結ばれて構成されている
  - 薬を選択する、地図を表示する、電話番号を閲覧する、など
- Mashlet が Glue で結ばれることで、Mashed-up Application が構成される。
- Mashed-up Application のグラフ構造が与えられた時に、ユーザの目的を達成するようなナビゲーションを提案するアルゴリズムを示すことが本論文の目的



（論文1）“Navigating in Complex Mashed-Up Applications”

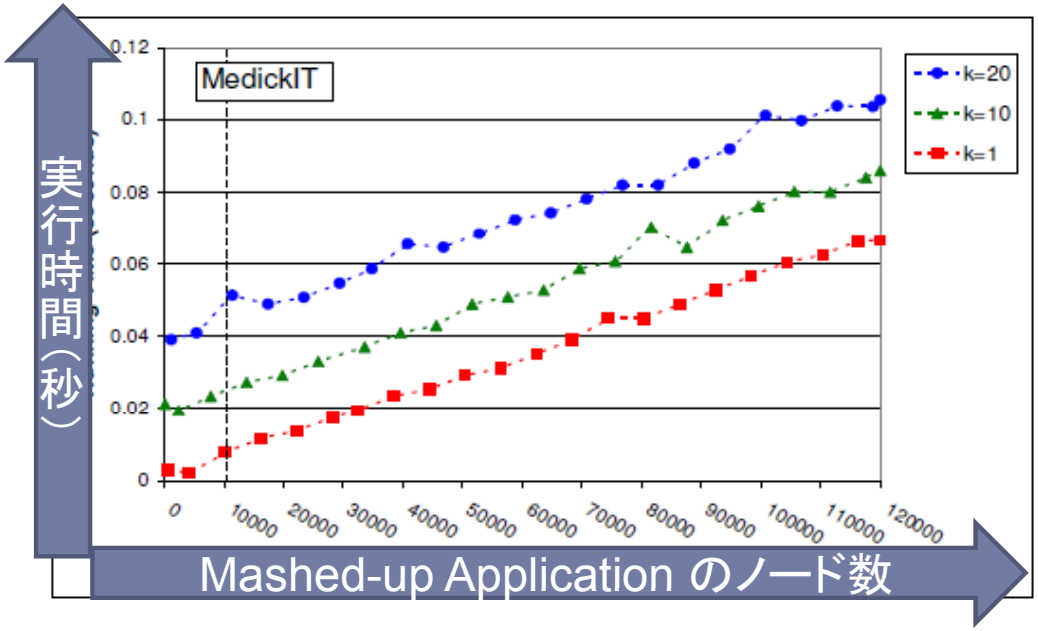
- システム利用の流れ
  1. （システム）Mashed-up Application の Flow Map を提示
  2. （ユーザ）　ユーザがクエリを指定
  3. （システム）ユーザの目的を達成することができる Top-k のナビゲーションフローを提案
  4. （ユーザ）　ユーザが選択、結果を確認

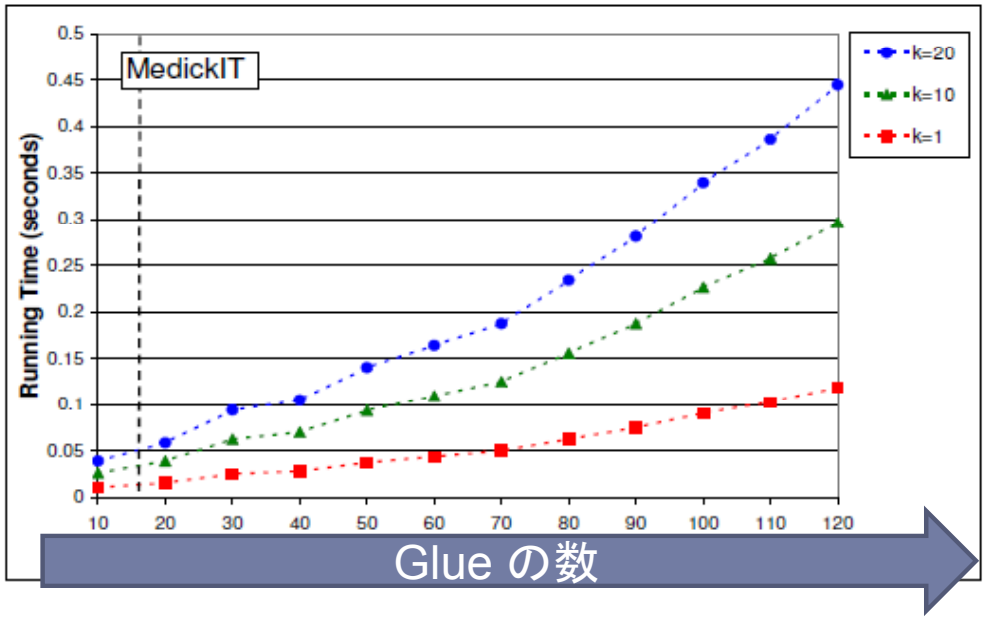
- 重み付け関数を定義することで Top-k を定義
  - 目的を達成するために必要なクリック数や入力回数、人気度など
- 最適なナビゲーションフローを求めることは NP完全 だが、現実には効率よく解ける



(論文1) “Navigating in Complex Mashed-Up Applications”

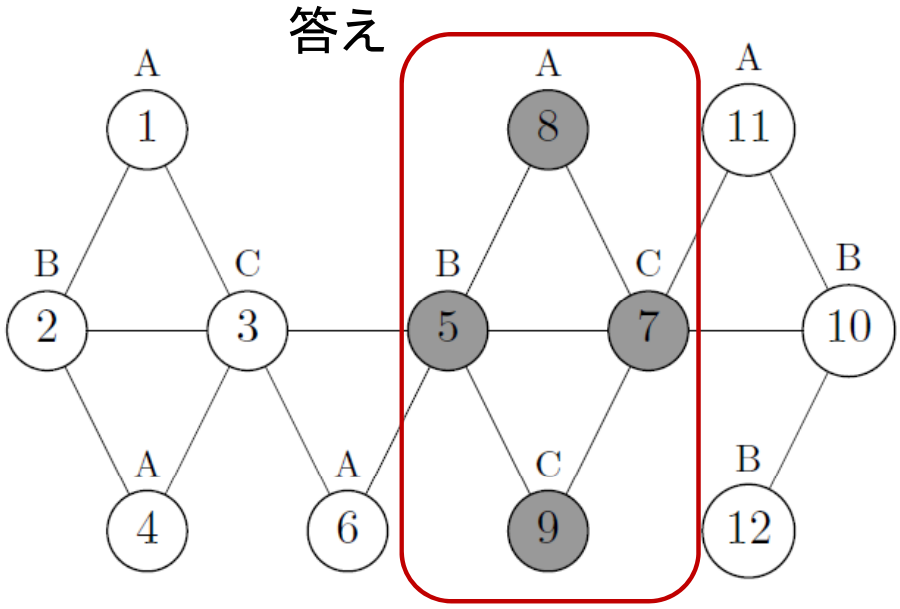
- **アルゴリズム実行速度の評価 （下図）**
  - Mashed-up Application の大きさ（ノード数）での評価 （下図左）
  - Glue の数で評価 （下図右）
- **実環境での評価**
  - 実際の Mashed-up Application を用いて、3つのタスクを指示
    - (1) 仕事場に一番近い医者の情報を探せ、(2) (1)を行い、その医者の情報を自分の携帯にメールで送れ、(3) 指定された患者の薬使用量グラフを閲覧せよ
  - 提案システムを用いることで、完了までの時間が平均 71% 短縮した
  - アルゴリズムの実行時間は下図の MedickIT の線で示されている





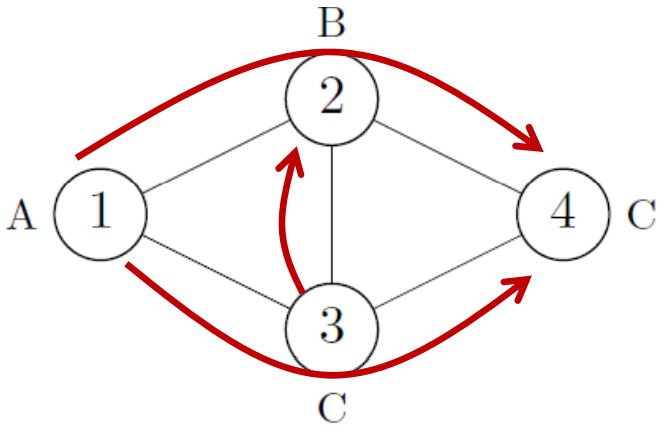
（論文2）“On Graph Query Optimization in Large Networks”

- **対象**
  - ノードにラベルがついた（無向）グラフ G
- **問題**
  - クエリとしてグラフ Q が与えられたとき、Q と同一の部分グラフを G から全て探す。ラベルも同一でなければならない
- この問題は NP完全であり、本論文では最短路に注目することで効率を高めたインデキシング方法 SPath を提案
  - クエリ Q をカバーする経路を用いてマッチングすることで、効率よく解くことができる



グラフ G

Q をカバーする経路の例



クエリ Q

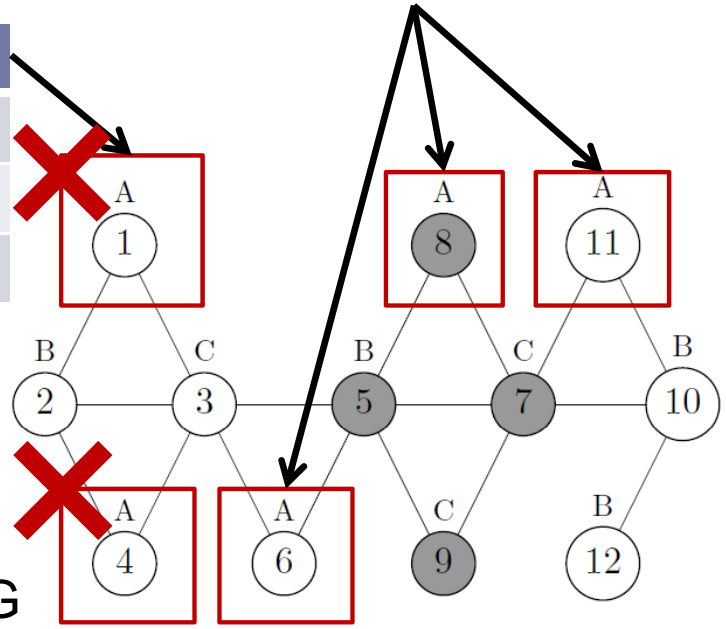
（論文2）“On Graph Query Optimization in Large Networks”

- アルゴリズムの概要
  - k_0 ホップ以下で辿れるノードに注目して候補ノードを枝狩り
  - 枝狩り後の候補ノード数が一番少ないクエリQのノード v* を起点として選択
  - Qを全てカバーする経路を、v* を始点とする経路から再帰的に探索し、v*に対する候補ノードごとにグラフGと経路をマッチング
    - どのような優先度で経路を選択すればよいかコスト関数を定義

クエリ Q の ノード1 に対する枝狩り後の候補ノード

k	k-Distance Set
0	A{1}
1	B{2}, **C{3}**
2	A{4, 6}, B{5}

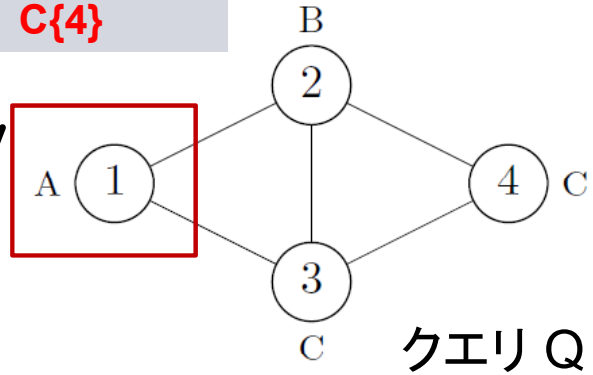
ラベルCについてクエリQの方が数が多いので、ノード1 は候補にならない



グラフ G

k	k-Distance Set
0	A{1}
1	B{2}, **C{3}**
2	**C{4}**

（注）ノード1 は v* として適切ではない。ノード2, 3の方が候補ノードは少ない



クエリ Q

（論文3）“Dremel: Interactive Analysis of Web-Scale Datasets”

- Google による分散処理フレームワークの発表

- Web 関連の開発では、解析を走らせて結果を確認して、また解析してといったインタラクティブな解析が必要になることが多い
  - （発表者補足）MapReduce は実装時に考慮するべき問題が多く、インタラクティブな解析に向いていないという議論が多い
- SQL-like なクエリを分散実行できる Dremel を提案
  - 単体で動作するが、MapReduceの置き換えではなく役割を補完

> MapReduceが出力したデータに対して
>
> ```
> DEFINE TABLE t AS /path/to/data/*
> SELECT TOP(signal1, 100), COUNT(*) FROM t
> ```
>
> のような SQL-like クエリを実行することを想定

- 入れ子構造のデータをカラムストア方式で保持し、SQL-like なクエリで大規模なデータに対する分散処理を実現

（論文3）“Dremel: Interactive Analysis of Web-Scale Datasets”

レコード1つ分の定義

```
message Document {
    required int64 DocId;
    optional group Links {
        repeated int64 Backward;
        repeated int64 Forward;
    }
    repeated group Name {
        repeated group Language {
            required string Code;
            optional string Country;
        }
        optional string Url;
    }
}
```

Example

```
DocId: 10
Links
    Forward: 20
    Forward: 40
    Forward: 60
Name
    Language
        Code: 'en-us'
        Country: 'us'
    Language
        Code: 'en'
    Url: 'http://A'
Name
    Url: 'http://B'
Name
    Language
        Code: 'en-gb'
        Country: 'gb'
```

```
DocId: 20
Links
    Backward: 10
    Backward: 30
    Forward: 80
Name
    Url: 'http://C'
```

- ネスト構造を許す
- 修飾子
  - required: 必須項目
  - repeated: 繰り返し可能＋省略可能
  - optional: 省略可能

（論文3）“Dremel: Interactive Analysis of Web-Scale Datasets”

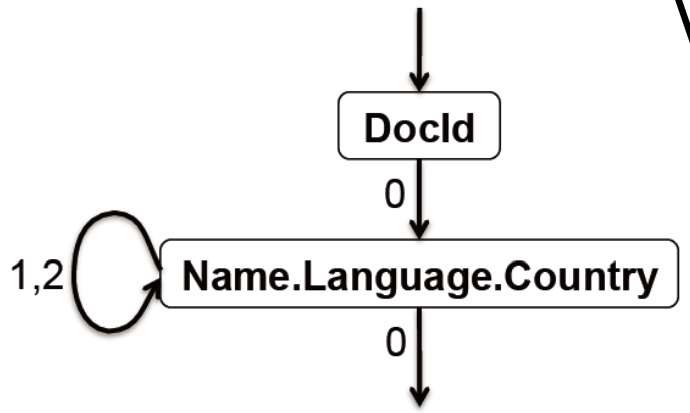
- 入れ子のデータ構造をカラムストア方式で保存・復元するアルゴリズムを提案
  - 単純にレコードごとに保存しただけは一意に保存できないので付加情報を付与
  - ステートマシンでの復元を実現
- カラムストアで保存することで、必要な項目のみを読みだすことができる
  - カラムストアの特徴である I/O の削減を実現

DocId

Value	r	d
10	0	0
20	0	0

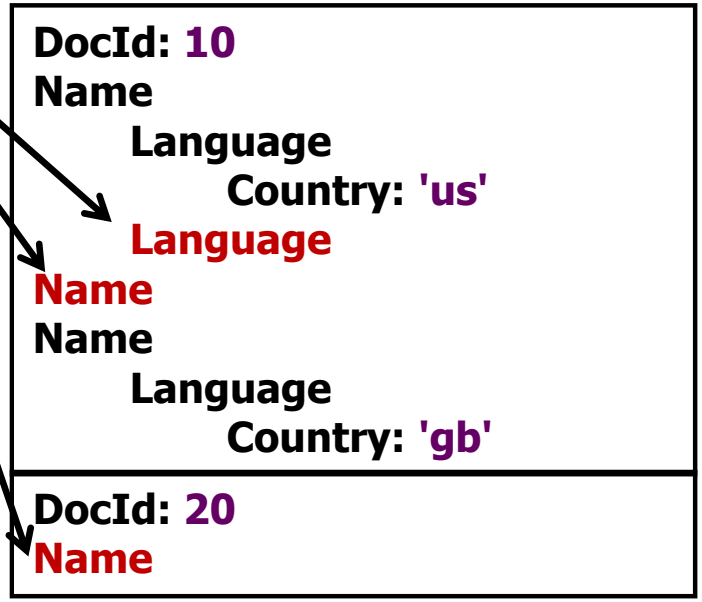
Name.Language.Country

Value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1



ノードのラベルで指定されたテーブルから
データを読み、次の行の r で状態遷移

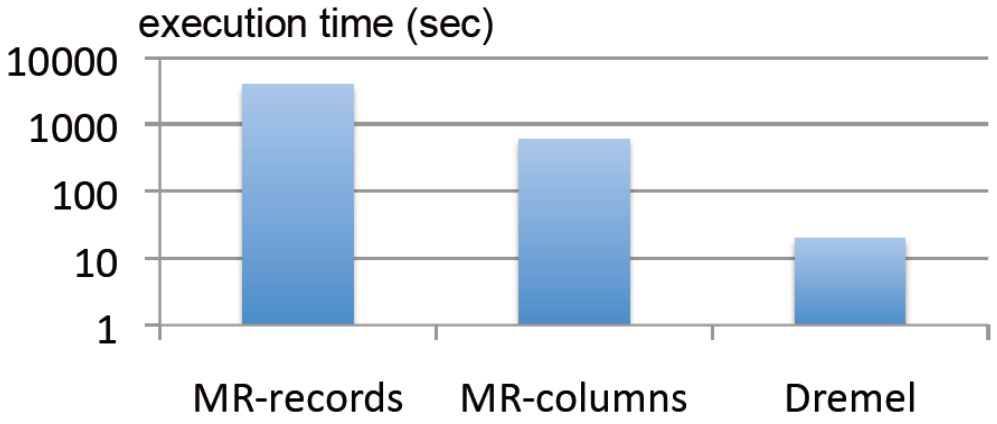
Name.Language.Country が省略されている

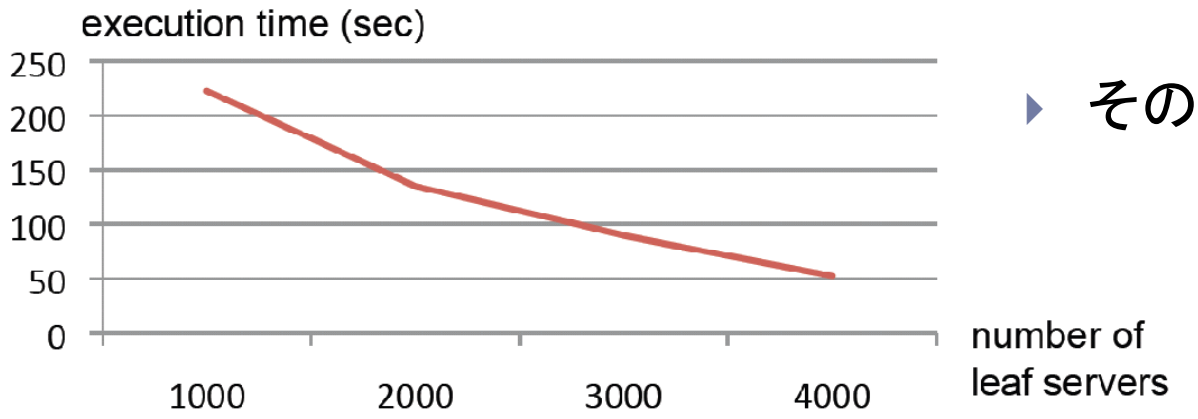


 ※図は論文より引用

（論文3）“Dremel: Interactive Analysis of Web-Scale Datasets”

Table name	Number of records	Size (unrepl., compressed)	Number of fields	Data center	Repl. factor
T1	85 billion	87 TB	270	A	3×
T2	24 billion	13 TB	530	A	3×
T3	4 billion	70 TB	1200	A	3×
T4	1+ trillion	105 TB	50	B	3×
T5	1+ trillion	20 TB	30	B	2×





- 実験データ（左上）
  - 様々な規模で5種類
  - データセンターの列やフィールドの多さに注目
- T1（850億ページ？）に対する単語数の平均を求める（左中央）
  - 10秒程度で完了
  - MapReduce の100倍単位の高速化
- 4000台のマシンまでほぼ線形なスケール（左下）
  - 対象データは T4
- その他多数の実験データ