【ICDE2014勉強会】

Session 2: Distributed Processing

担当：渡辺陽介（名古屋大学）

Session2の論文リスト

1. R-Store: A Scalable Distributed System for Supporting Real-time Analytics
   - Feng Li, M. Tamer Ozsu, Gang Chen, and Beng Chin Ooi (National Univ. Singapore, Univ. Waterloo, Zhejiang Univ.)

2. Blazes: Coordination Analysis for Distributed Programs
   - Peter Alvaro, Neil Conway, Joseph M. Hellerstein, David Maier (UC Berkeley, Portland State Univ.)

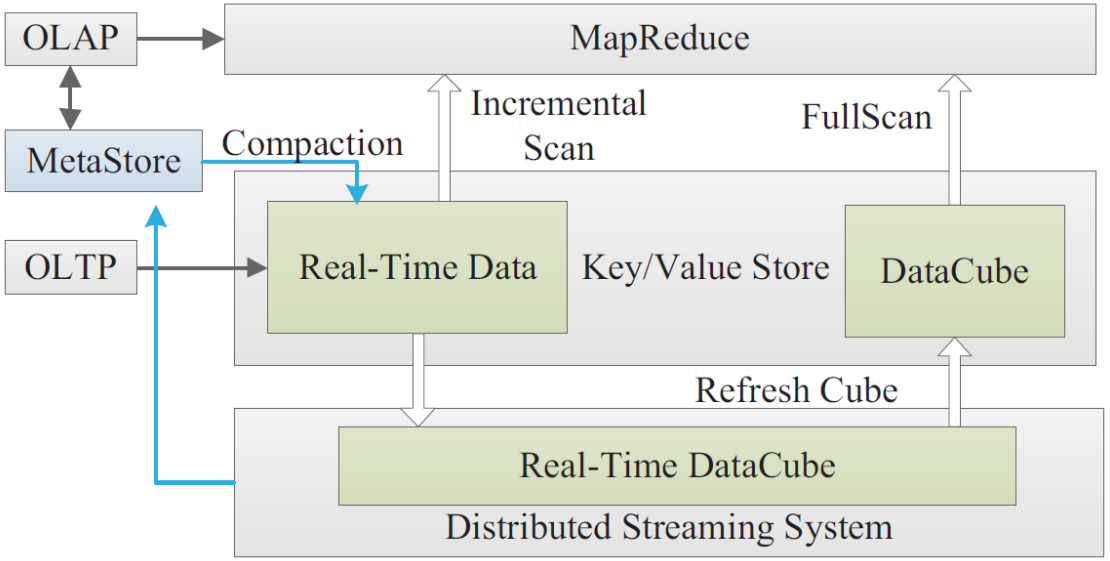
3. Query Optimization of Distributed Graph Pattern Matching
   - Jiewen Huang, Kartik Venkatraman, Daniel J. Abadi, (Yale University)

[1本目] R-Store: A Scalable Distributed System for Supporting Real-time Analytics

- 概要
  - OLTPクエリを扱うシステムと，OLAPクエリを扱うシステムは，伝統的に分かれており，OLAP側へデータをロードすることが必要だった
  - OLTPクエリとOLAPクエリの両方を扱うことができ，最新の更新データに対してデータキューブ上の分析処理が行えるシステムR-Storeの提案

Figure1より引用

- 貢献
  - HBase, Streaming Map/Reduce の統合
  - HBaseから更新差分をとるIncrementalScanを実装
  - コストモデル

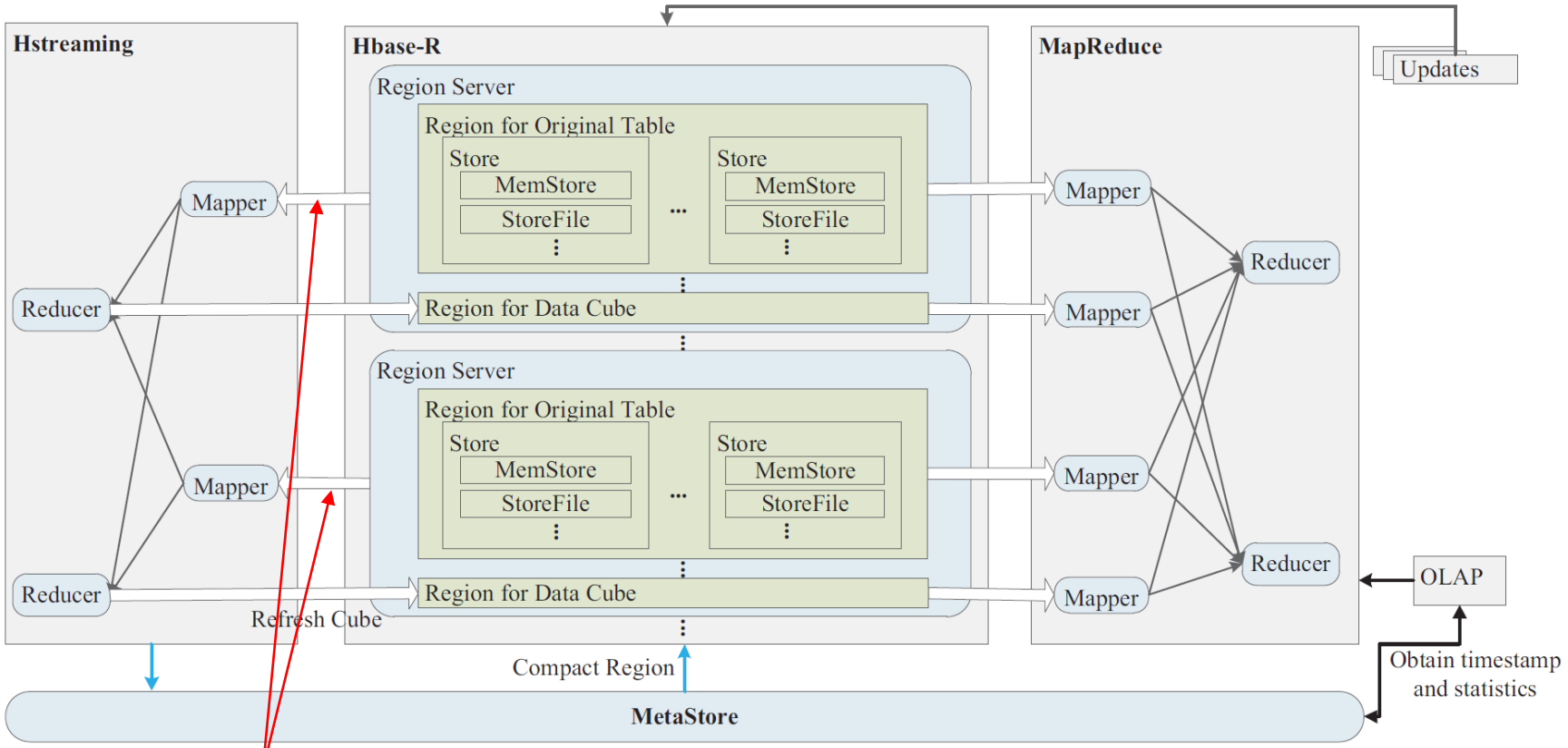


[1本目] R-Storeにおけるデータフロー

ストリーム処理:
更新データのキューブへの反映

Key-Value Store:
OLTP用の表管理
データキューブの管理

OLAP処理:



キューブ構築後の差分だけを取得する
IncrementalScanを使用

Figure2より引用

[1本目] 評価実験

- 構成
  - Hbase 40ノード，MapReduce 40ノード，Hstreaming 20ノード
  - CPU Intel X3430 2.4GHz, RAM 8GB, HDD 2x500GB
- TPC-H (既存キーの更新が発生するように改変)

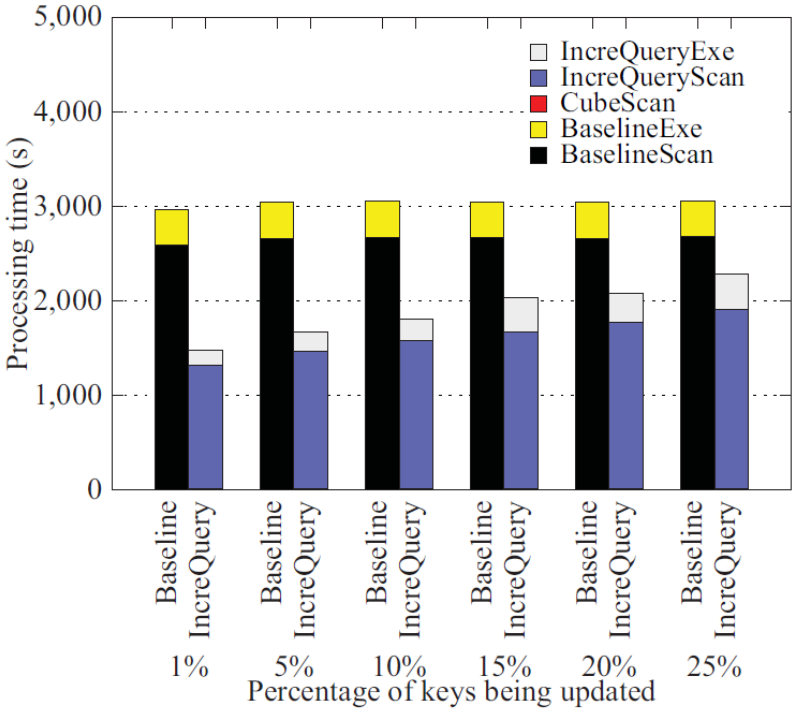


Figure8より引用

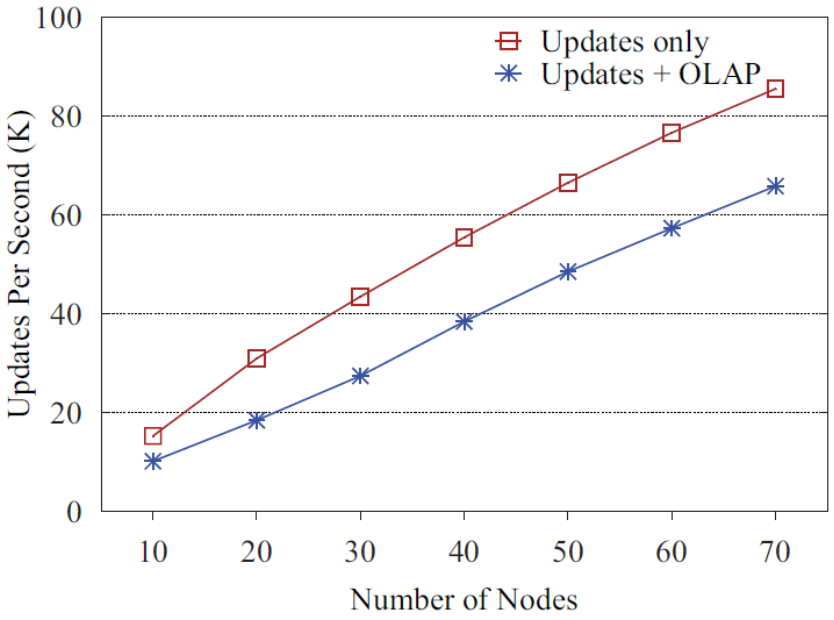
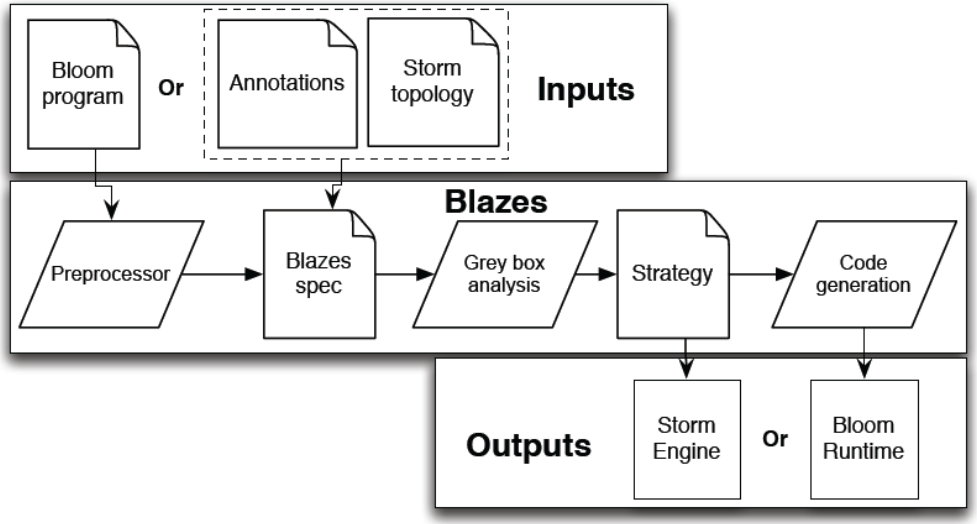


Figure11(a)より引用

[2本目] Blazes: Coordination Analysis for Distributed Programs

- 概要
  - 協調動作を行う分散システムでは，データの整合性を保証することが必要
    - 障害時の挙動をデバッグすることはプログラマの負担が高い
  - 分散処理プログラムの解析ツールBlazesの提案
    - 協調動作の必要なプログラムを識別
      - Twitter Storm(ストリーム処理システム)向けのデータフロー
      - Bloom(プログラム言語)で書かれたアプリケーション
    - 適切な協調のためのコードを生成

Figure1
より引用

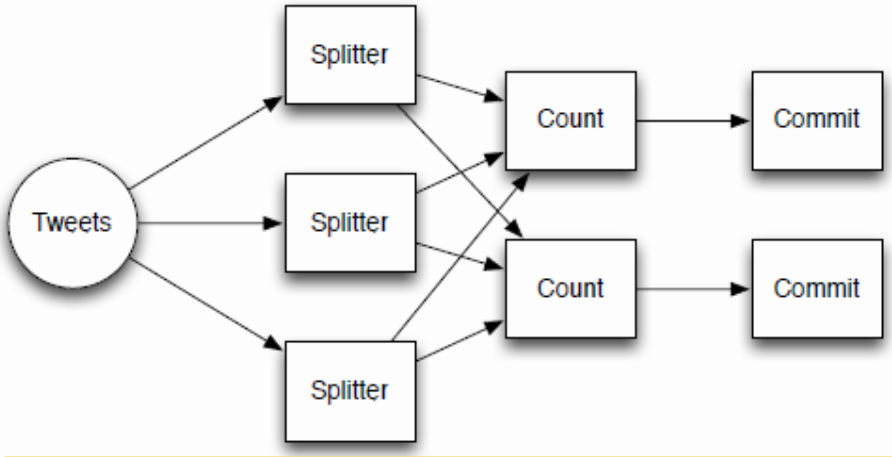


担当：渡辺（名大）

[2本目]

- 異常(Anomalies)のクラス
  - Nondeterministic ordering (Async)
    - タプルの出力順が決定的でない（再送した時に順番が変わりうる）
  - Cross-run nondeterminism (Run)
    - コンポーネントが実行する度に同じ入力に対して異なる出力を生成
  - Cross-instance nondeterminism (Inst)
    - 複製されたインスタンス上で実行されたコンポーネントが同じ入力に対して異なる結果を出力
  - Replica divergence (Diverge)
    - 複数のレプリカが永久に不整合な状態
- 性質：Confluence
  - あるコンポーネントがすべての入力順序に対して同じ出力を生成する（扱いが楽）
  - Non-confluenceなものをどうするか？

アノテーション



Twitter Stormのデータフロー
(word count)
Figure2より引用

```
Splitter:
    annotation:
        - { from: tweets, to: words, label: CR }
Count:
    annotation:
        - { from: words, to: counts, label: OW,
        subscript: [word, batch] }
Commit:
    annotation: { from: counts, to: db, label: CW }
```

コンポーネントへのアノテーション
Figure7より引用

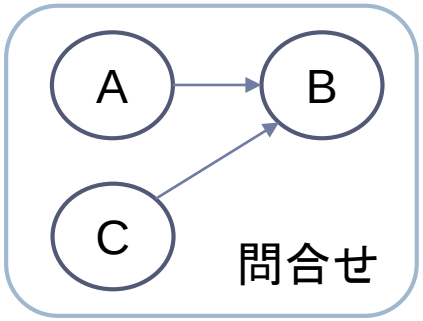
Severity	*Label*	*Confluent*	*Stateless*
1	CR	**X**	**X**
2	CW	**X**	
3	OR_{gate}		**X**
4	OW_{gate}		

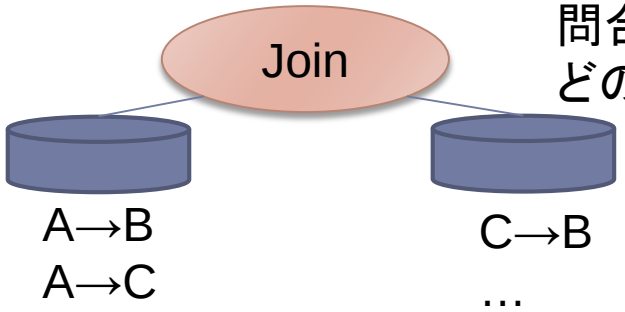
C : Confluent (出力が決定的)
O : non-confluent (非決定性あり)
R : Read-only (内部状態なし)
W : Write states (内部状態あり)

項書き換えにより出力の性質を導出し，適切な協調を選択
- Sealing strategies: ウインドウごとのメッセージ同期など
- Ordering strategies: メッセージ順序を保証するサービスの利用

[3本目] Query Optimization of Distributed Graph Pattern Matching

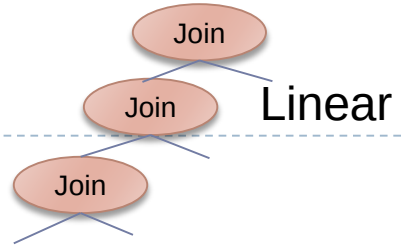
- 概要
  - 部分グラフのパターンマッチング処理を分散で実行する場合の問合せ最適化手法を提案
- 前提
  - 分散配置
    - グラフ内の各頂点をハッシュ値により分散配置
    - 有効辺は始点側の頂点と同じ場所に配置
  - 部分グラフ問合せ処理手順
    - 問合せとなるグラフも頂点ごとに部分問合せに分割
    - 部分問合せの結果をJoinにより統合する



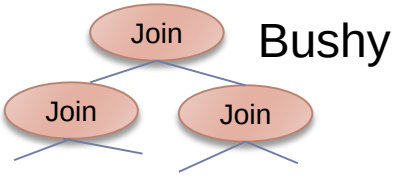


問合せのグラフが複雑化した時にどの順でJoinを計算するのか

[3本目] 論文の貢献

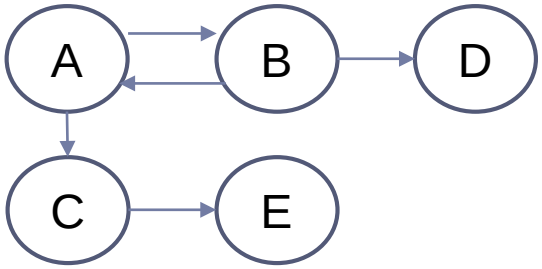


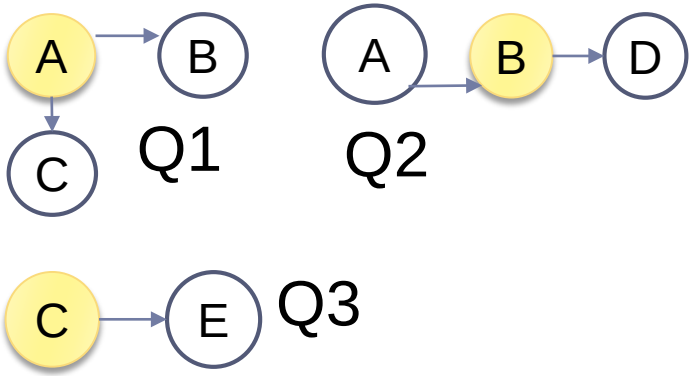
- 動的計画法に基づく最適化アルゴリズム
  - Linear plan
  - bushy plan

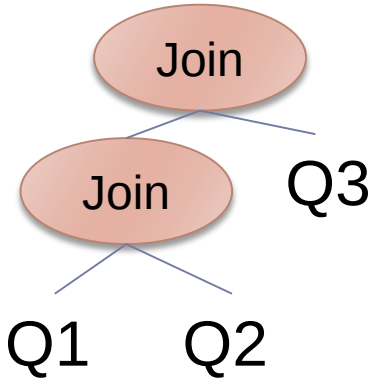


- cycleを利用した中間結果の削減
  - 問合せにcycleがある場合はサイクル部分を優先的に結合する
  - cycleの条件は制約が強いので，中間結果を減らせる傾向がある
- 計算結果の再利用による同一の部分グラフの問合せの削減

Example 3 (Linear plan)







[3本目] 評価実験

- データセット
  - Amazon product graph (上)
  - Youtube video graph (中)
  - Web graph (下)
- 比較対象
  - Greedy(baseline)
  - DP-linear
  - DP-bushy
  - Cycle-greedy
  - Cycle-bushy

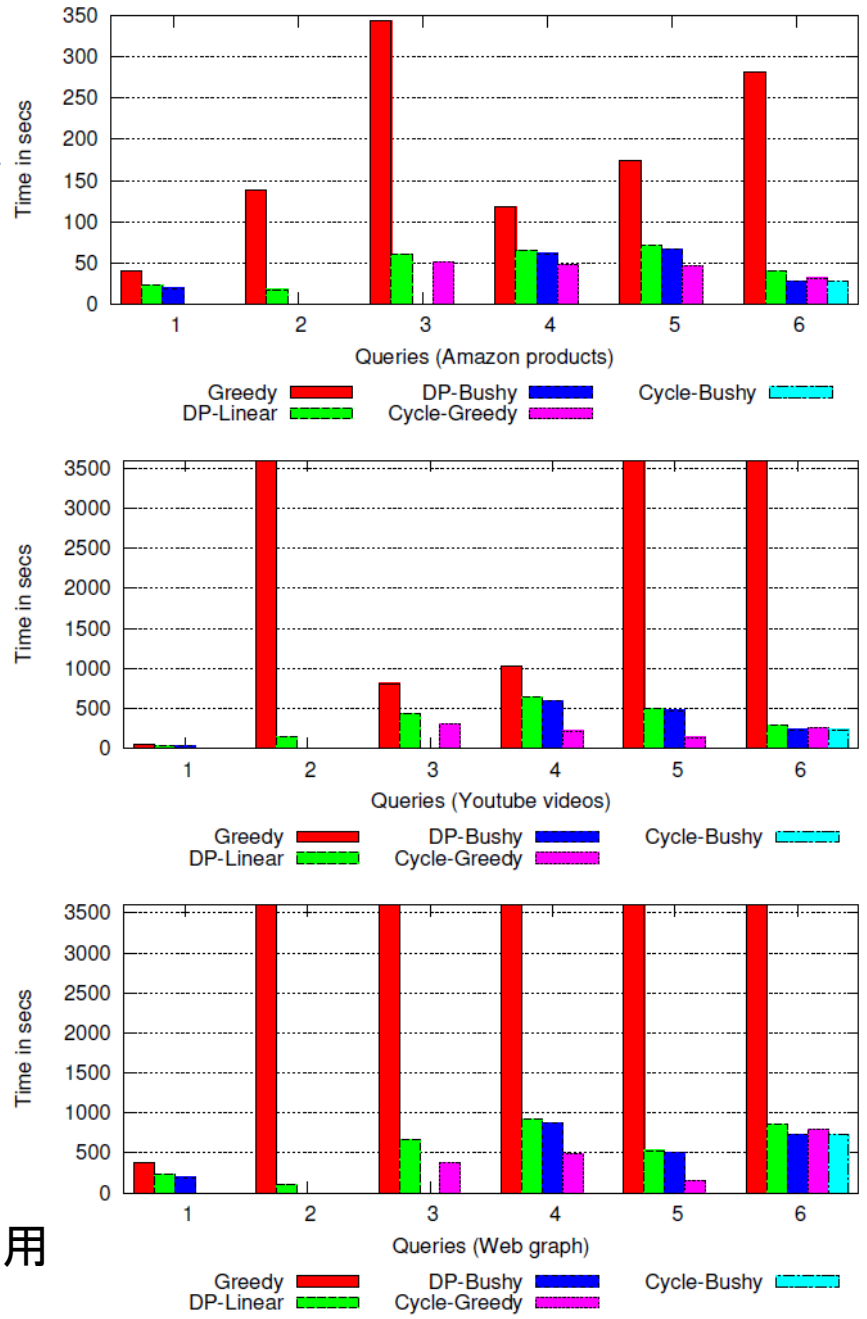


Figure12より引用