【SIGMOD2010勉強会】

Session4: Data Streams & Time-series Data

担当：渡辺陽介（東京工業大学）

PR-Join: A Non-Blocking Join Achieving Higher Early Result Rate with Statistical Guarantees

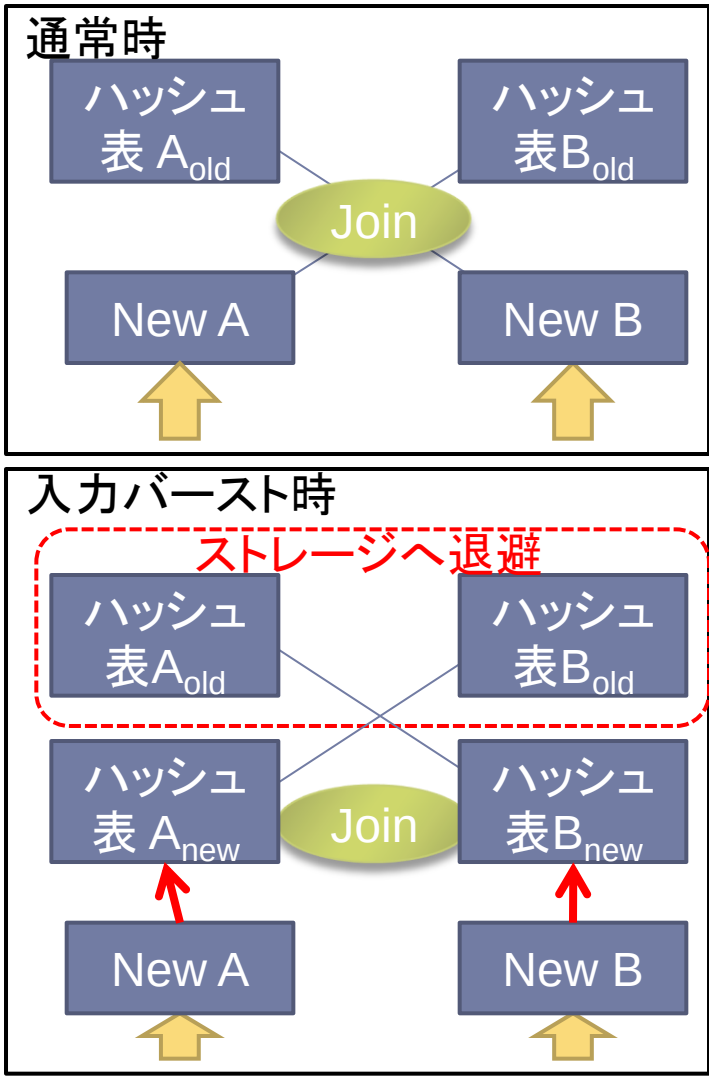
- Shimin Chen, Intel Labs Pittsburgh; Phillip Gibbons, Intel Labs Pittsburgh; Suman Nath, Microsoft
- 概要
  - Early resultを高速に生成可能なJoinアルゴリズムの提案
    - データウェアハウス等でインタラクティブな分析をする場合，完全でなくてもいいので集約処理の途中結果を見たい要求がある
  - 既存のJoinアルゴリズムの問題点
    - GRACE hash join [4,16] & sort-merge join
      - 処理全体のIOコストは低いが，early result生成までは時間がかかる
    - Ripple join [10]
      - Early resultを生成が早いが，完全な結果を出すまでのIOコストが高い

→early result生成が早く，IOコストが低いJoinの提案

  - Partitioned Expanding Ripple Join (PR-Join)

Ripple Join

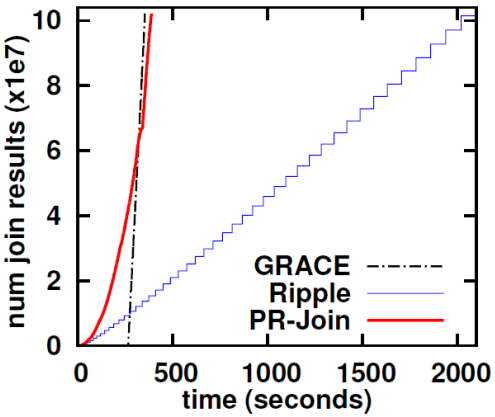
- 通常時 (symmetric hash join)
  1. Aの新しいタプルtをハッシュ表に追加
  2. タプルtでBのハッシュ票をprobe
- 入力タプルでメモリが埋まりそうな場合
  1. ハッシュ表A_{old},B_{old}をストレージへ退避
  2. AとBの新しいタプルのハッシュ表を構築
  3. A_{old}をスキャンし，B_{new}をprobe
  4. B_{old}をスキャンし，A_{new}をprobe
  5. A_{new}をスキャンし，B_{new}をprobe

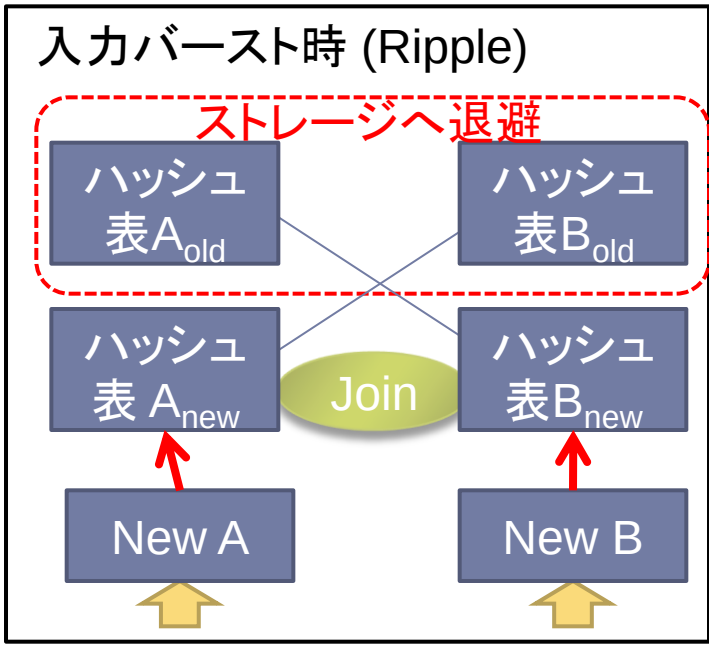


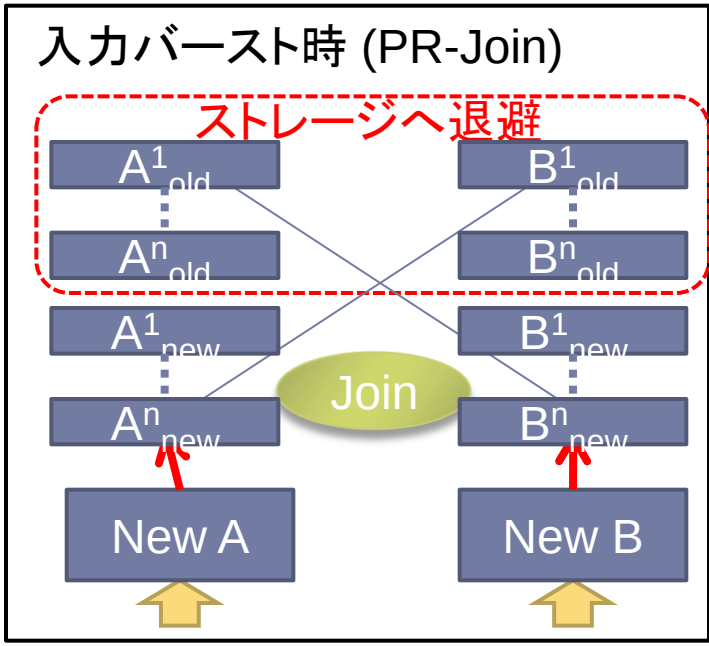
PR-Join

- 通常時 (symmetric hash join)は同じ
- 入力タプルでメモリが埋まりそうな場合
  1. ハッシュ表A_{old},B_{old}をn個に分割して，ストレージへ退避
  2. AとBの新しいタプルについてもn個のハッシュ表を構築
  3. For(int i=1; i <= n ; i++)
     1. A^i_{old}をメモリ上に読みだす
     2. B^i_{old}とB^i_{new}をスキャンし，A^i_{old}とA^i_{new}をprobe

Figure2







PODS: A New Model and Processing Algorithms for Uncertain Data Streams

- Thanh Tran, UMass Amherst; Liping Peng, UMass Amherst; Boduo Li, UMass Amherst; Yanlei Diao, University of Massachusetts; Anna Liu, UMass Amherst
- 概要
  - 不確実性データストリームにおいて，連続確率変数を扱うことのできるシステムPODSの提案
    - 離散的なPossible world modelでは可能性の数え上げが大変
  - データモデル
    - 各属性の確率分布をGaussian Mixture Model (GMM)で表す
      - 既存手法 Histgram [12], weighted particles [18]
  - 集約演算(sum, avgなど)
    - Gaussianの性質を使って多重積分をせずに済ますために近似を行う
  - Join演算
    - Equi-join, inequality-join

PODS Data Model

- スキーマ: $A^d \cup A^p_i \cup A^p_j$
  - A^d: 決定的な値をとる属性知の集合
  - A^p_i : 連続的で不確実な値をとる属性知の集合 (互いに独立)
  - A^p_j : 連続的で不確実な値をとる属性知の集合 (相関あり)
- タプル t の確率分布

$$f_{\mathbf{X}}(\mathbf{x}) = \prod_i f_i(x_i) \prod_j f_j(\mathbf{x}_j),$$

  - $f_i(x_i)$: A^p_i のGaussian Mixture distribution
  - $f_j(\mathbf{x}_j)$: A^p_j をベクトルxとした多変数Gaussian Mixture distribution

Aggregation and Join

T1 Object Location

Tag id	Loc	Prob
0x333	10	0.5
	20	0.5

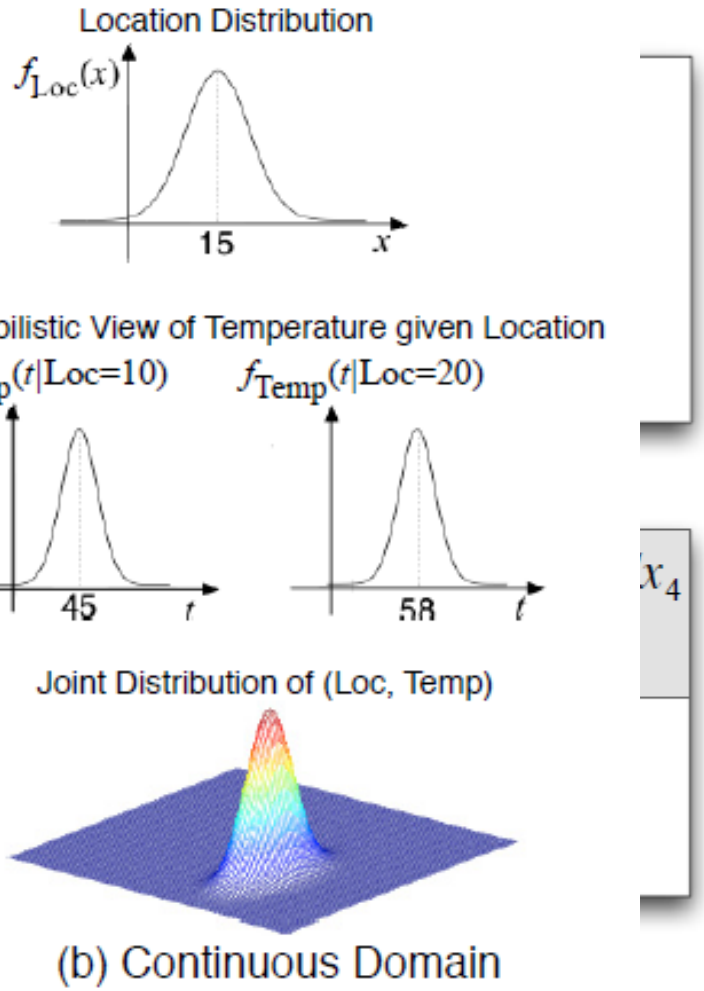
T2 Temperature

Loc	Temp	Prob
10	30	0.2
	50	0.8
20	50	0.6
	70	0.4

T3 Possible Worlds

PW	Tag id	Loc	Temp	Prob
1	0x333	10	30	0.1
2	0x333	10	50	0.4
3	0x333	20	50	0.3
4	0x333	20	70	0.2

(a) Discrete Domain



(b) Continuous Domain

Figure 3 Figure 5

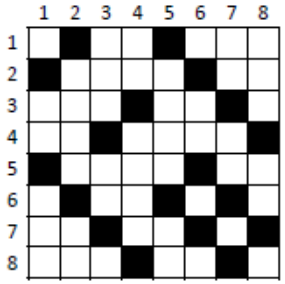
Fast Approximate Correlation for Massive Time-series Data

- Abdullah Mueen, UC Riverside; Suman Nath, Microsoft; Jie Liu, Microsoft Research
- 概要
  - n種類の時系列データ(長さm)から，閾値Tより大きな相関値をもつ時系列データのペアを高速に発見し，相関を計算する手法
  - データセンターにおけるサーバ監視アプリ
    - 1サーバあたり500個のパフォーマンスカウンタが存在
    - サーバ間の依存関係の調査や，負荷分散の分析をしたい
  - 全組合せの相関係数を計算するコストは高い
    - 相関がほとんどない時系列のペアに興味はない
    - 誤差が保証できれば，正確な値でなくてもよい

→閾値Tにより探索範囲を減らし，近似により許容誤差ε で計算

手法1: Threshold Correlation Matrix

1. 時系列データS_i $(0 \leq i \leq n)$に離散フーリエ変換をかける
   - DTF[i] : 変換後の第k番目までの係数
   - DFTの間の距離が一定値以下の時系列ペアだけ残す
     - distance(DTF[i], DTF[j]) ≤ sqrt(2m(1−T))
2. 残った組合せについて，相関係数を計算
   - ただしキャッシュサイズの制約から，時系列データをメモリにロードする順番によってIOコストが変化する
   - グラフ分割の最適化問題として，F−M algorithm[6] を用いて解く



(a) Pruning Matrix

Signals read	In Cache	Computed pairs
1, 2, 3, 4	1, 2, 3, 4	(1, 2), (3, 4)
5, 6, 7, 8	5, 6, 7, 8	(5, 6), (6, 7), (7, 8)
1, 2	1, 2, 5, 6	(1, 5), (2, 6)
3, 4, 7, 8	3, 4, 7, 8	(3, 7), (4, 8)

(b) Caching Strategy 1, 14 disk reads

Signals read	In Cache	Computed pairs
1, 2, 5, 6	1, 2, 5, 6	(1, 2), (1, 5), (2, 6), (5, 6)
7	2, 5, 6, 7	(6, 7)
3, 4, 8	3, 4, 7, 8	(3, 4), (3, 7), (4, 8), (7, 8)

(c) Caching Strategy 2, 8 disk reads

Figure 2

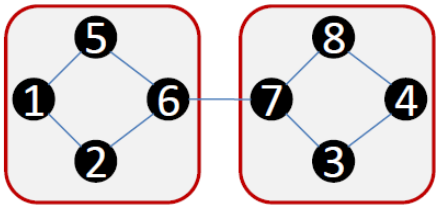
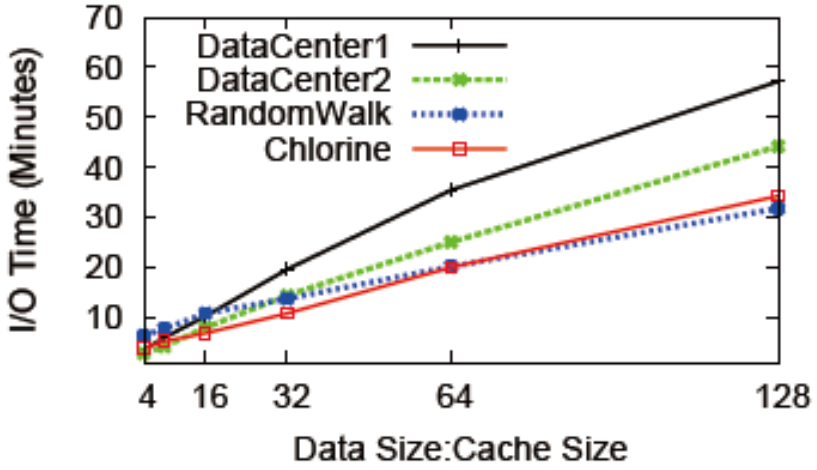


Figure 3

評価実験

- 実験データ
  - DataCenter1: 350台のサーバの1日分のTCPコネクション, 11200種類の時系列データで, 長さは2880
  - DataCenter2: 120台のサーバのCPU使用率. 4745種類, 長さ2880
  - Chlorine[17]: 配水システムにおける塩素濃度監視.166種類,長さ2155
  - RandomWalk: 人工データ. 16384種類, 長さは2880



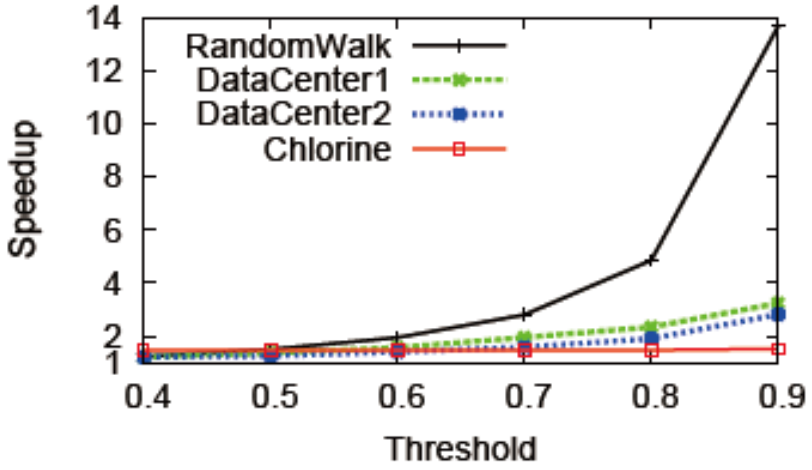


Figure 5 (b) Impact of cache size (c) CPU cost reduction

An Algorithmic Approach to Event Summarization

- Peng Wang, Fudan University; Haixun Wang, Microsoft; Majin Liu, Fudan University; Wei Wang, Fudan University
- 背景
  - システムログのイベント系列からサマリを生成
    - サマリ：ある区間のイベント発生分布を説明する統計モデル
  - 従来方式
    - ある固定長のセグメントに区切って，セグメントごとにサマリ(モデル)を生成
    - 問題点：モデル間の関係が説明されない

→システムの内部状態の遷移を，モデル間の遷移として記述したい

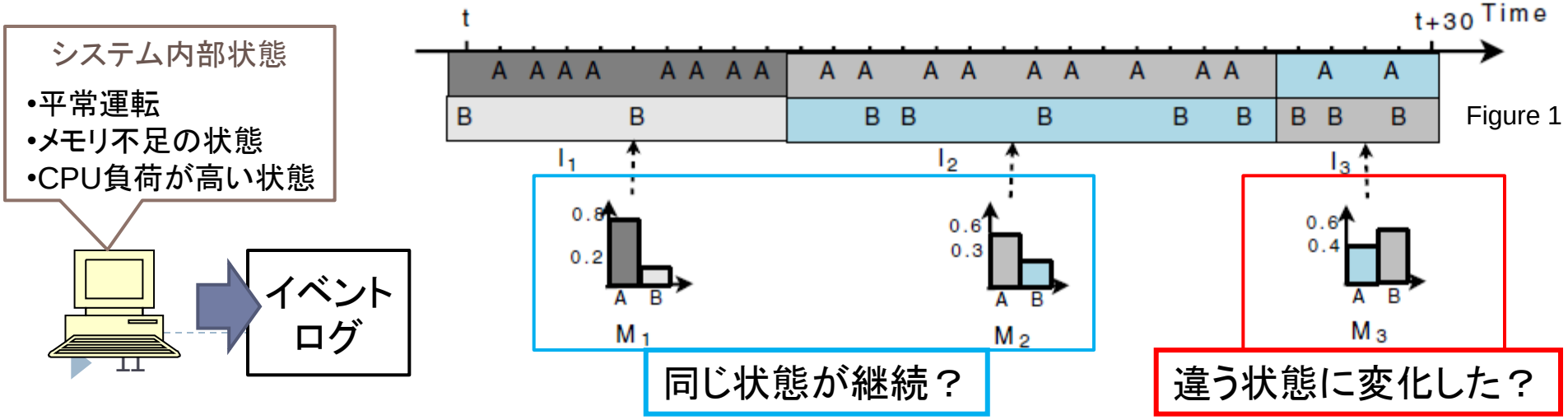


Figure 1

An Algorithmic Approach

- システムの内部状態ごとにモデルが変化すると仮定し，モデル間の状態遷移を推測して，サマリとして提示する

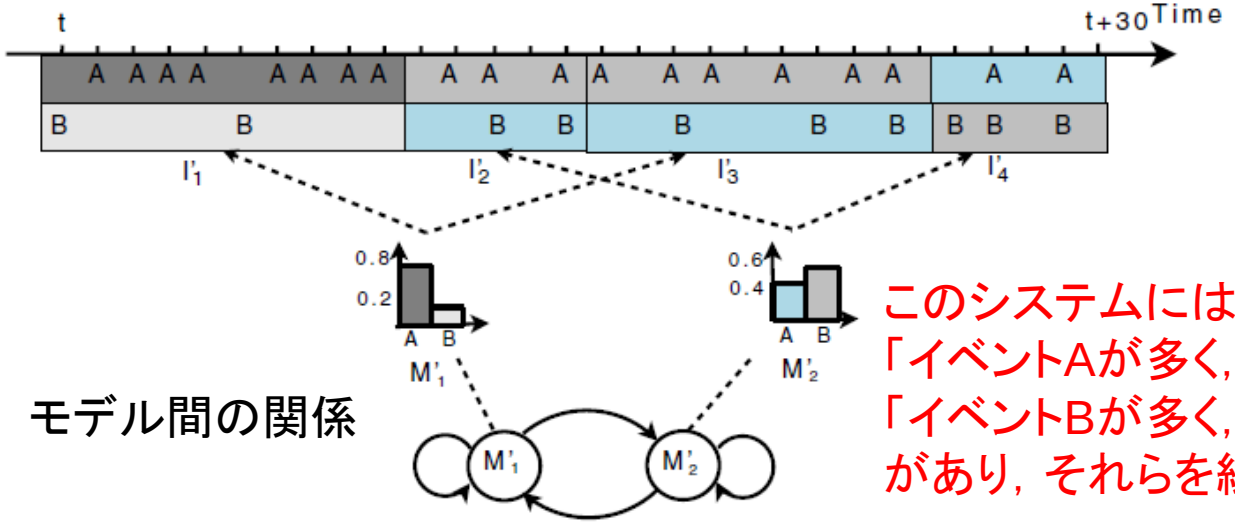


Figure 2

モデル間の関係

このシステムには
「イベントAが多く，Bが少ない状態」(I_1, I_3)と
「イベントBが多く，Aが少ない状態」(I_2, I_4)
があり，それらを繰り返している

- 問題設定：「状態数をKとして，各状態のモデルと，対応するイベント系列上のセグメントを求める」
- Hidden Markov Modelに基づく学習アルゴリズム
  - M_{IND}: イベントの発生に排他的な関係等がない場合
  - M_{DEP}: イベントの発生に排他的な関係等がある場合