Scalable Parallelization of Skyline Computation for Multi-core Processors

堀 川　　隆

概要

- 趣旨
  - Skyline Computationを並列実行するアルゴリズムを提案，評価
- 提案アルゴリズム
  - Q-Flow : データ分割は単純な方法（データをsortして分割）で行い，各々を並列に処理 （アルゴリズムの骨子説明用？）
  - Hybrid : pivot となるデータを選び，それに基づいてデータ分割を行った上で，Q-Flowと同様の手法で並列処理
- ポイント
  - 独立して実行できるDominance test （２点を比較し，片方が優勢かどうかを判定）を並列に実行
  - Dominance testの実行回数を削減

アルゴリズム (Q-Flow)

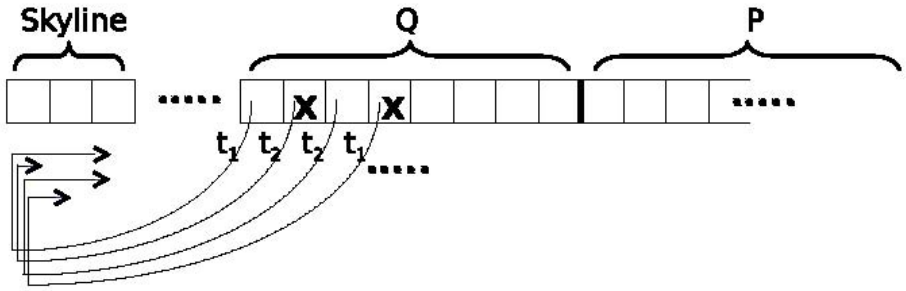
Algorithm 1 Q-Flow $(\mathcal{P}, \alpha \longrightarrow \mathrm{SKY}(\mathcal{P}))$

1: $S \leftarrow \emptyset$.
2: Prefilter, sort, and/or partition $\mathcal{P}$. ▷ Initialization
3: **while** $\mathcal{P} \neq \emptyset$ **do**
4: $Q \leftarrow$ next α points of $\mathcal{P}$.
5: $\mathcal{P} \leftarrow \mathcal{P} \setminus Q$.
6: **for** $i \in [0, |Q|)$ **do** ▷ Parallel Phase I
7: **if** $\exists q \in S : q \prec Q[i]$ **then**
8: Mark $Q[i]$ as pruned.
9: Remove pruned $Q[i]$ from Q. ▷ Compression
10: **for** $i \in [0, |Q|)$ **do** ▷ Parallel phase II
11: **if** $\exists j \in [0, i) : Q[j] \prec Q[i]$ **then**
12: Mark $Q[i]$ as pruned.
13: Remove pruned $Q[i]$ from Q. ▷ Compression
14: Append Q to S. ▷ Update structure
15: **return** S.

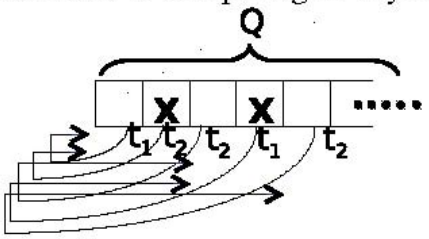
← データをQ, P, …に分割

← 各点を既判明のSkylineとDT

← 残った点をブロック内の点とDT



(a) Q-Flow Phase I: comparing to skyline points.



(b) Q-Flow Phase II: comparing to other survivors.

（図は総て論文から引用，以下同様）

アルゴリズム (Hybrid)

Algorithm 1 Q-Flow $(\mathcal{P}, \alpha \longrightarrow \mathrm{SKY}(\mathcal{P}))$

$S \leftarrow \emptyset$.
Prefilter, sort, and/or partition $\mathcal{P}$. ▷ Initialization
while $\mathcal{P} \neq \emptyset$ **do**
  $Q \leftarrow$ next α points of $\mathcal{P}$.
  $\mathcal{P} \leftarrow \mathcal{P} \setminus Q$.
  for $i \in [0, |Q|)$ **do** ▷ Parallel Phase I
    if $\exists q \in S : q \prec Q[i]$ **then**
      Mark $Q[i]$ as pruned.
  Remove pruned $Q[i]$ from Q. ▷ Compression
  for $i \in [0, |Q|)$ **do** ▷ Parallel phase II
    if $\exists j \in [0, i) : Q[j] \prec Q[i]$ **then**
      Mark $Q[i]$ as pruned.
  Remove pruned $Q[i]$ from Q. ▷ Compression
  Append Q to S. ▷ Update structure
return S.

Algorithm 2 updateS&M$(S, \mathcal{M}(S), Q)$

Append Q to S.
Pop sentinel off of $\mathcal{M}(S)$.
$(m, i) \leftarrow \mathrm{top}(\mathcal{M}(S))$
for $j \in [0, |Q|)$ **do**
  if $Q[j].\mathrm{m} = m$ **then**
    $Q[j].\mathrm{m} \leftarrow \mathrm{part}(Q[j], S[i])$.
  else
    $(m, i) \leftarrow (Q[j].m, |S| + j)$.
    Push (m, i) onto $\mathcal{M}(S)$.
Push sentinel $(2^d, |S| + |Q|)$ onto $\mathcal{M}(S)$.

Algorithm 3 compareToSky $(S, \mathcal{M}(S), q \longrightarrow \{\mathrm{true}, \mathrm{false}\})$

for $i \in [0, |\mathcal{M}(S)|)$ **do**
  $m \leftarrow \mathcal{M}(S)[i].\mathrm{m}$.
  if m is not incomparable to $q.\mathrm{m}$ **then**
    $s \leftarrow \mathcal{M}(S)[i].\mathrm{s}$; $t \leftarrow \mathcal{M}(S)[i+1].\mathrm{s}$
    $m' \leftarrow \mathrm{part}(q, S[s])$.
    if $m' = 2^d - 1 \wedge q \not\equiv S[s]$ **then return** true.
    for $j \in (s, t)$ **do**
      if $S[j].m$ is not incomparable to m' **then**
        if $S[j] \prec q$ **then return** true.
return false.

Algorithm 4 compareToPeers $(Q, \mathrm{me} \longrightarrow \{\mathrm{true}, \mathrm{false}\})$

$i \leftarrow 0$.
while $Q[i].|m| < Q[\mathrm{me}].|m|$ **do**
  if $Q[i]$ is not pruned **then**
    if $Q[i].m$ is not incomparable to $Q[\mathrm{me}].m$ **then**
      if $Q[i] \prec Q[\mathrm{me}]$ **then return** true.
  $i \leftarrow i + 1$.
while $Q[i].m < Q[\mathrm{me}].m$ **do**
  $i \leftarrow i + 1$.
while $i < me$ **do**
  if $Q[i] \prec Q[\mathrm{me}]$ **then return** true.
return false.

評価

- 比較対象
  - Pskyline : the state-of-the-art multicore algorithm
  - BSkyTree : the state-of-the-art sequential algorithm
  - PBSkyTree : BSkyTreeを筆者らが並列化したもの

- 環境
  - マシン
    - CPU : Xeon E5-2670(8 cores) x 2，Hyper Threading off ⇒ 16 cores
    - メモリ: 64 GB
  - データセット
    - Standard skyline data generator により，correlated, independent, anticorrelated の3種類を生成
      - Dimension: 4 ~ 16 (default 12)
      - Cardinality: 500K ~ 8M (default 1M)
    - 実データ（参考用）: NBA, House (比較的小規模) と Weather(大規模)

結果（実行時間）

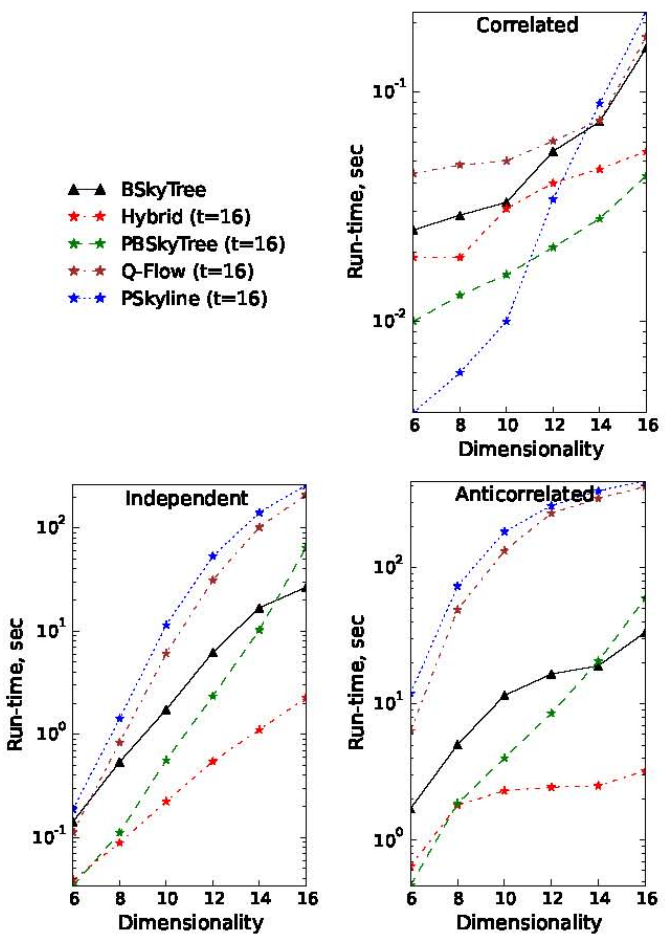


Fig. 5: State-of-the-art performance w.r.t. d ($n = 1$M).

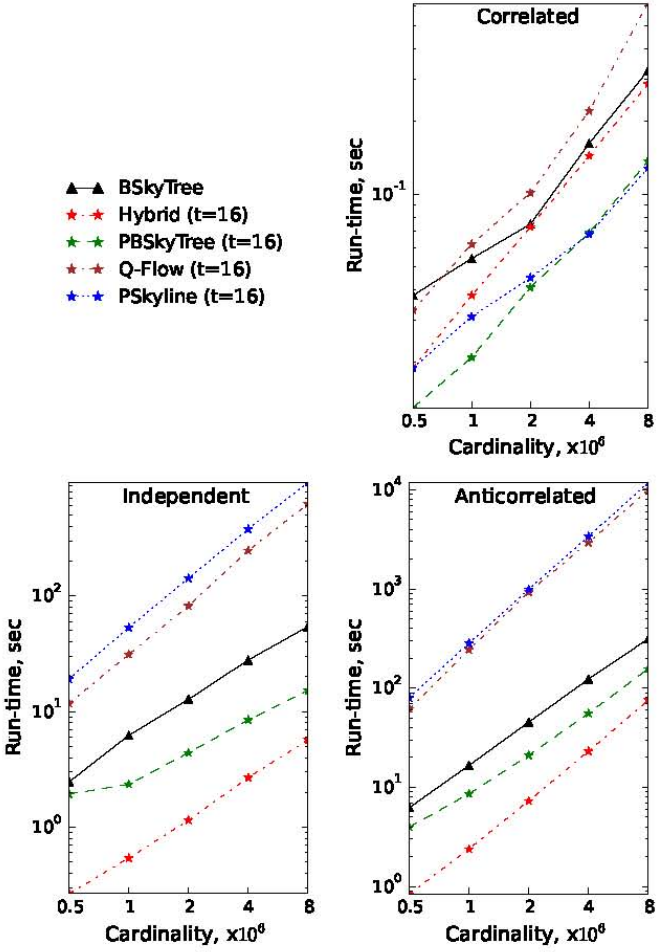


Fig. 6: State-of-the-art performance w.r.t. n ($d = 12$).

処理時間の増え方は，データの種類によって異なる

結果（実行時間のbreak down）

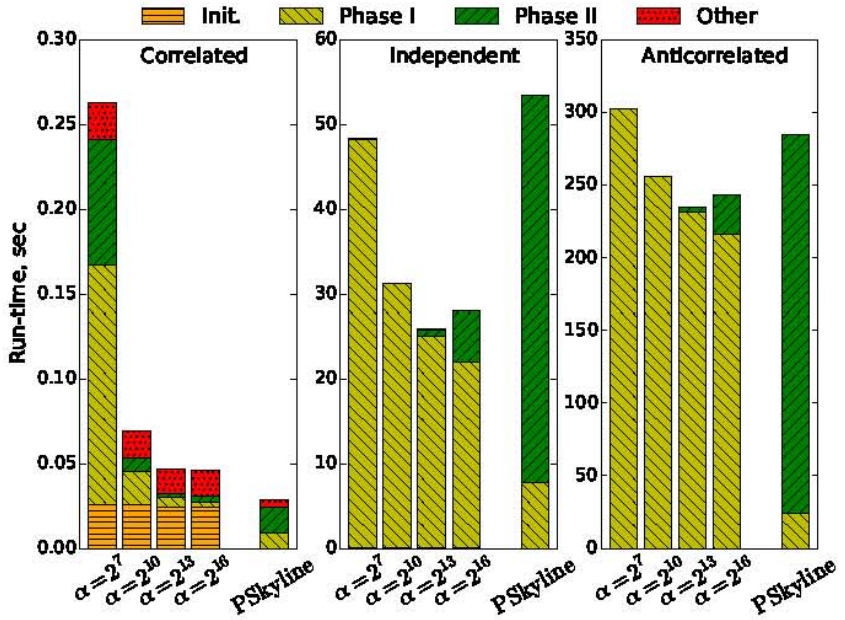


Fig. 7: Effect of α in Q-Flow with varied α ($n = 1$M, $d = 12$)

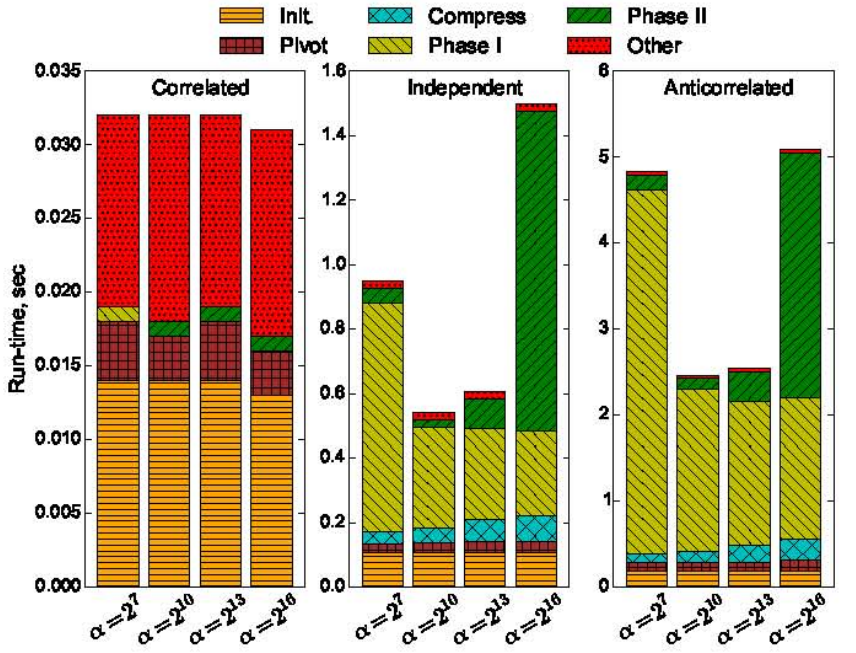


Fig. 8: Effect of α on Hybrid with varied α ($n = 1$M, $d = 12$)

- Correlatedでは初期化時間の割合が大きい．他のデータでは，初期化時間の増加を補って余りある並列化効果が出ている．

まとめ

- Skyline Computationの並列アルゴリズムを考案
  - Q-Flow: 最先端の並列アルゴリズム（ Pskyline）を上回る
  - Hybrid: partitioningを行う版，これまでの内，最も効率的とされているアルゴリズム（BSkyTree），および，その並列化版（PBSkyTree）を上回る