Optimal Splitters for Temporal and Multi-version Database

Wangchao Le, Feifei Li, Yufei Tao, Robert Christensen

THE UNIVERSITY OF UTAH®

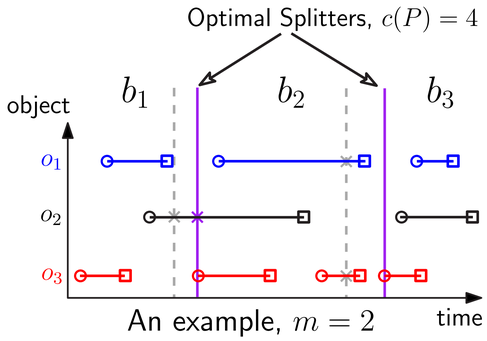
KAIST 한국과학기술원 SINCE 1971

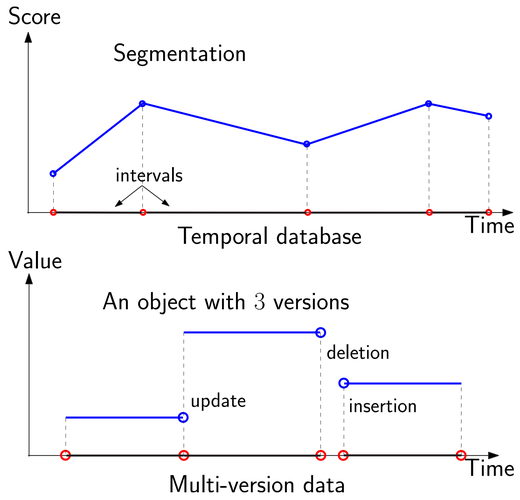
香港中文大學 THE CHINESE UNIVERSITY OF HONG KONG

Motivation and Problem Formulation

Temporal and multi-version data used extensively in:

- financial market
- scientific applications
- data warehousing





Baseline Method – Dynamic Programing

Given a splitter l and a set of intervals I stored in an array.

Algorithm cost: $O(kN^2)$

Internal Memory Method

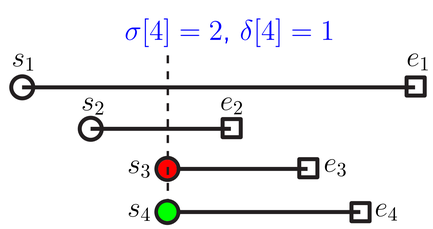
Sketch of Cost-t algorithm

1. Optimal cost t^* is in range of $\mathbf{R} = [1, N]$
2. Binary Search on $\mathbf{R}$
   - if t is infeasible, then any $t' < t$ is also infeasible
   - Solve $O(\log N)$ instances of Cost-t splitters problem
3. Report t^* as result when t^* is **feasible** but $t^* - 1$ is **infeasible**

Algorithm cost: $O(N \log N)$

Stabbing Count Array

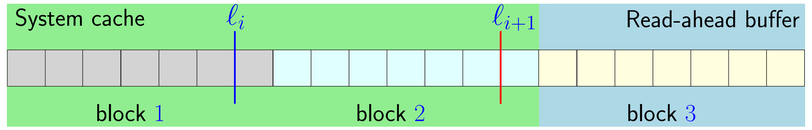
- Sort $I = \{[s_1, e_1] \dots [s_N, e_N]\}$
- The stabbing-count array for I
  - $\forall s_i \in I$, maintain two counts σ, δ
    - $\sigma[i] = |I^x(s_i)|$, # intervals strongly covering s_i
    - $\delta[i] = |I^o_<(s_i)|$, # intervals in $I^o(s_i)$ with ids less than i



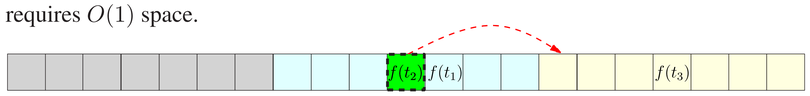
t-jump method

1. place splitters in ascending order
2. l_{i+1} is pushed as far as possible from l_i, let each new b_i have size t
3. if not achievable, move l_{i+1} backward just enough to form the new b_i

External Memory Method



- t-jump scans **forwardly**, next block to be read is **uniquely defined**.
- One execution requires $O(1)$ space.



- initialize h threads of cost-t tests $1 \le t_1 < t_2 < \ldots < t_h \le N$
- $f(t_i)$ the frontier of cost-t_i testing
- at any time activate the thread with $\min(f(t_i))$

Cost Analysis

sort disk-resident array of $O(N/B)$ blocks	$SORT(N) = (N/B)\log_{M/B}(N/B)$
construct the stabbing count array	$= O(SORT(N))$ I/Os
One round of Concurrent Cost-t testings	$= O(N/B)$ I/Os
# rounds of Concurrent Cost-t testings	$= O(\log_M N) \le O(\log_{M/B} N/B)$
Cost to find t^*	$= SORT(N)$ at most
Retrieve the optimal splitter	$= O(\min(k, N/B))$ I/Os

Experimental Results

	Internal	External
Data set	a subset of *Meme*	a subset of *Temp*
size	$\sim$ 21 MB	$\sim$ 4.1 GB
N	$\sim$ 1 million	$\sim$ 200 million
k	40	5000
h	not applicable	5

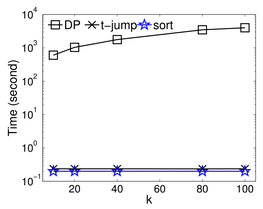


Figure 1: Running time of internal memory methods **vary** k

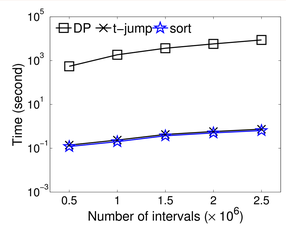


Figure 2: Running time of internal memory methods **vary** N

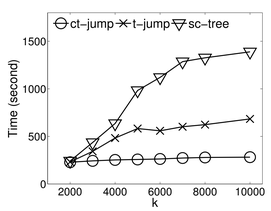


Figure 3: **Query time** of Queryable splitters **vary** k

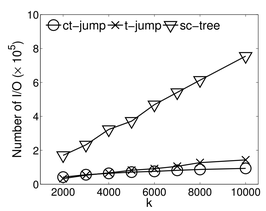


Figure 4: **Query IO** of Queryable splitters **vary** k

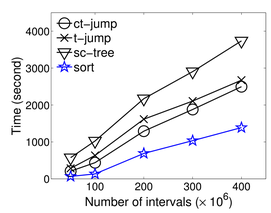


Figure 5: **Query time** of static splitter **vary** N

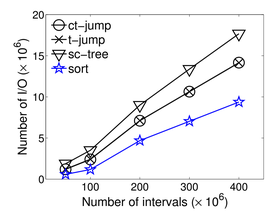


Figure 6: **Total IO** of static splitter **vary** N

Conclusion

Temporal and multi-version databases often generate massive amounts of data. Therefore, it becomes increasingly important to store and process this data in a distributed and parallel fashion. This makes an important contribution in solving the optimal splitters problem, which is essential in enabling efficient distributed and parallel processing of such data.