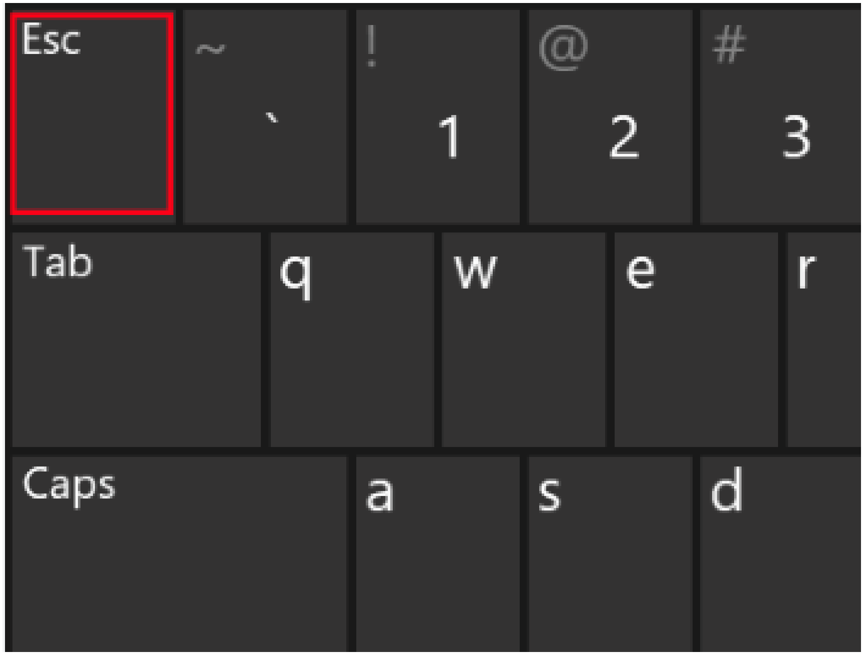
<u>**Esc**</u>: An Early-Stopping Checker for Budget-aware Index Tuning



Xiaoying Wang, **<u>Wentao Wu</u>**, Vivek Narasayya, Surajit Chaudhuri

Microsoft Research, Redmond, USA

Workload-Driven Index Tuning

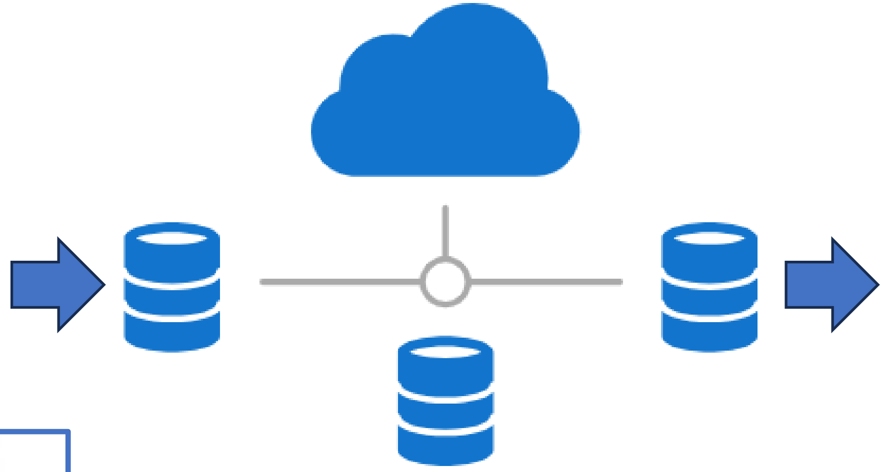
SQL Query Workload

```
SELECT l_shipmode,
SUM(case when o_orderpriority in ('1-URGENT','2-HIGH')
    then 1 else 0 end) as high_line_count,
SUM(case when o_orderpriority not in ('1-URGENT','2-HIGH')
    then 1 else 0 end) as low_line_count
FROM orders, lineitem
WHERE o_orderkey = l_orderkey
AND l_shipmode in ('[SHIPMODE1]', '[SHIPMODE2]')
AND l_commitdate < l_receiptdate
AND l_shi
AND l_rec
AND l_rec
GROUP BY
ORDER BY
```

```
SELECT c_name, c_custkey, o_orderkey, o_orderdate,
    o_totalprice, sum(l_quantity)
FROM customer, orders, lineitem
WHERE  o_orderkey IN ( SELECT  l_orderkey
    FROM  lineitem
    GROUP BY l_orderkey
    HAVING  sum(l_quantity) > [QUANTITY] )
AND c_custkey = o_custkey
AND o_or
GROUP BY
ORDER BY
```

```
SELECT n_name, c_custkey, ..., c_comment,
        sum(l_extendedprice*(1-l_discount))
FROM customer, orders, lineitem, nation
WHERE o_orderdate >= ['DATE']
AND o_orderdate < ['DATE'] + interval '3' month
AND l_returnflag = 'R'
AND o_orderkey = l_orderkey
AND c_custkey = o_custkey
AND c_nationkey = n_nationkey
GROUP BY n_name, c_custkey, ..., c_comment
ORDER BY revenue desc;
```

Database System



Index Configuration

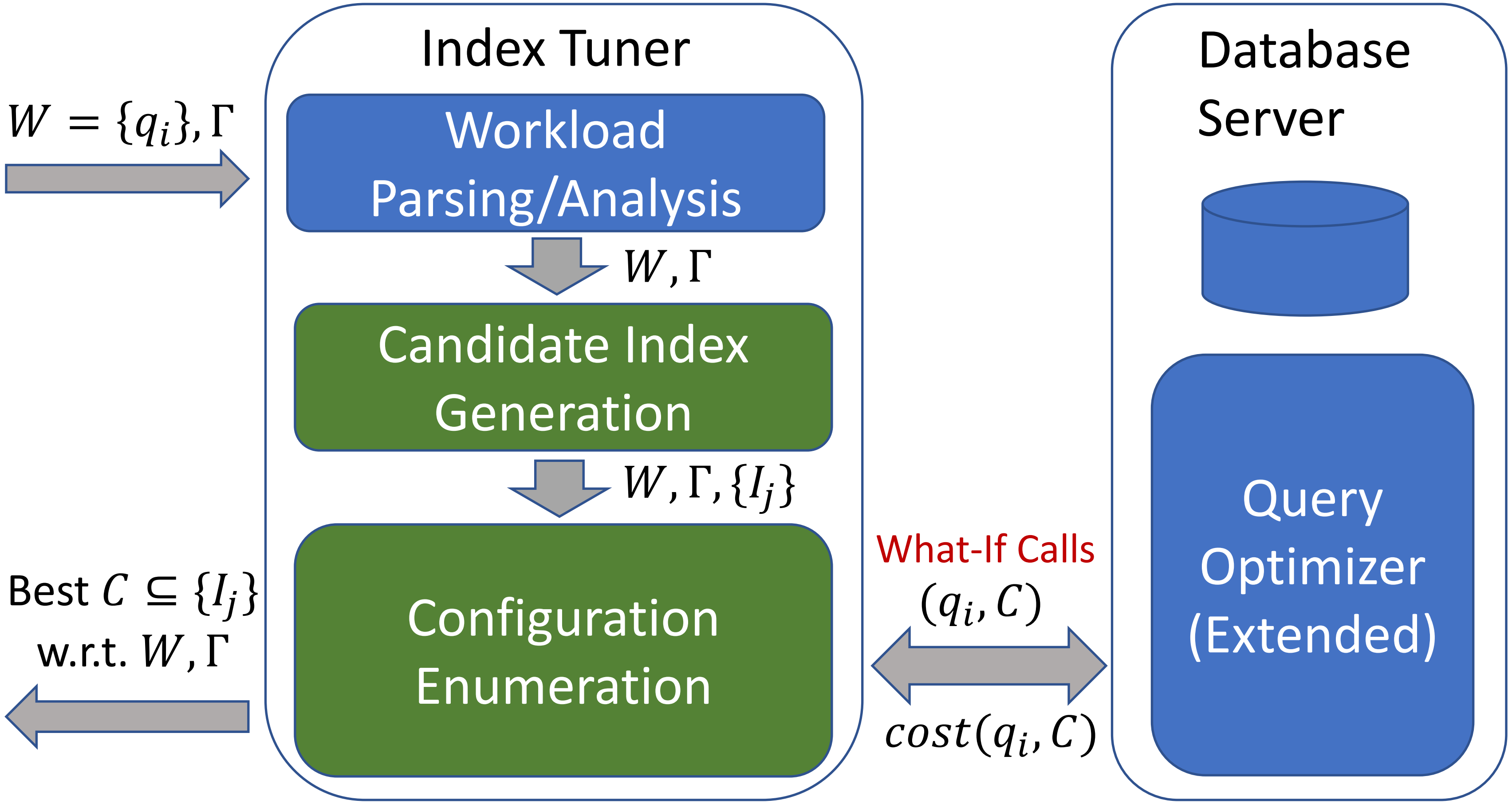
```
CREATE INDEX index_name
ON table_name (column 1, column 2, …);
```

```
CREATE INDEX index_name
ON table_name (column 1, column 2, …);
```

……

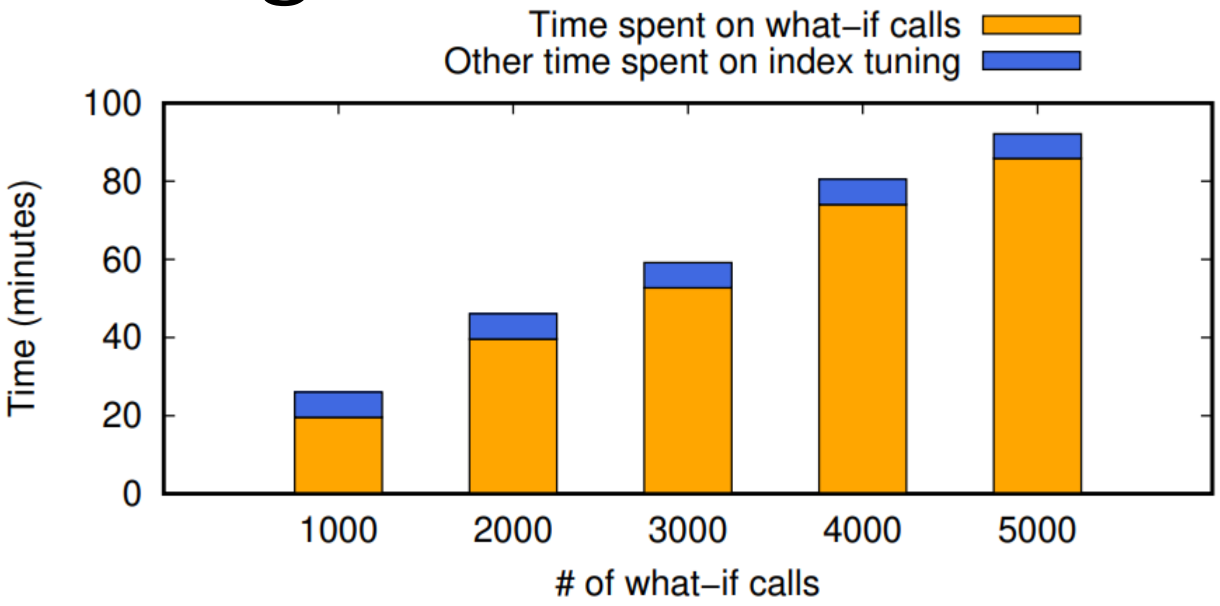
```
CREATE INDEX index_name
ON table_name (column 1, column 2, …);
```

The Architecture of An Index Tuner



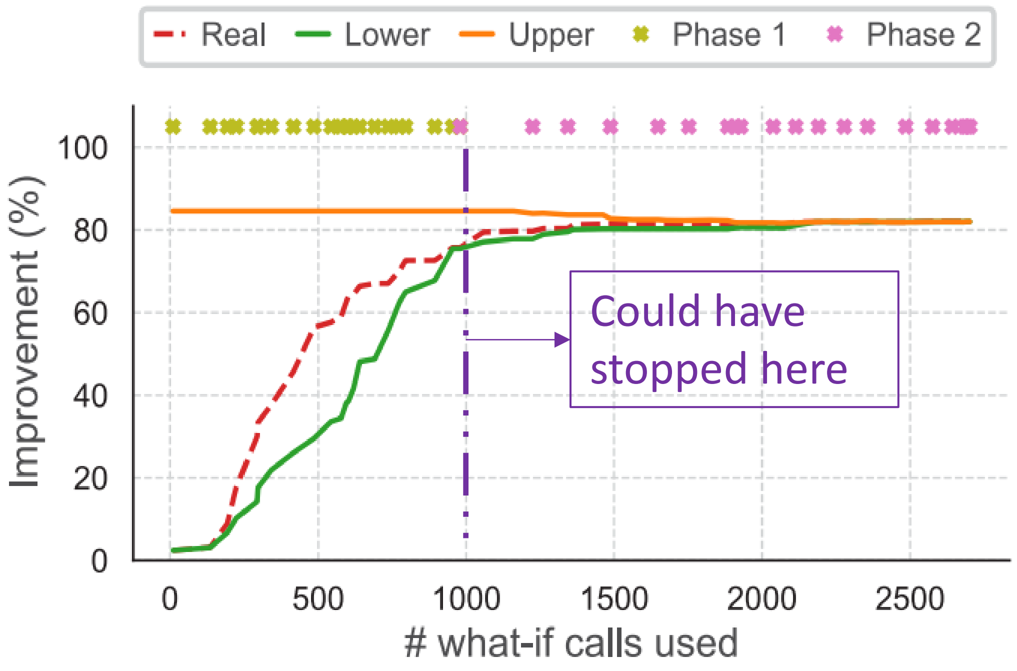
Γ represents a set of index tuning constraints, e.g., the max # of indexes, the max storage space, etc.

Budget-aware Index Tuning

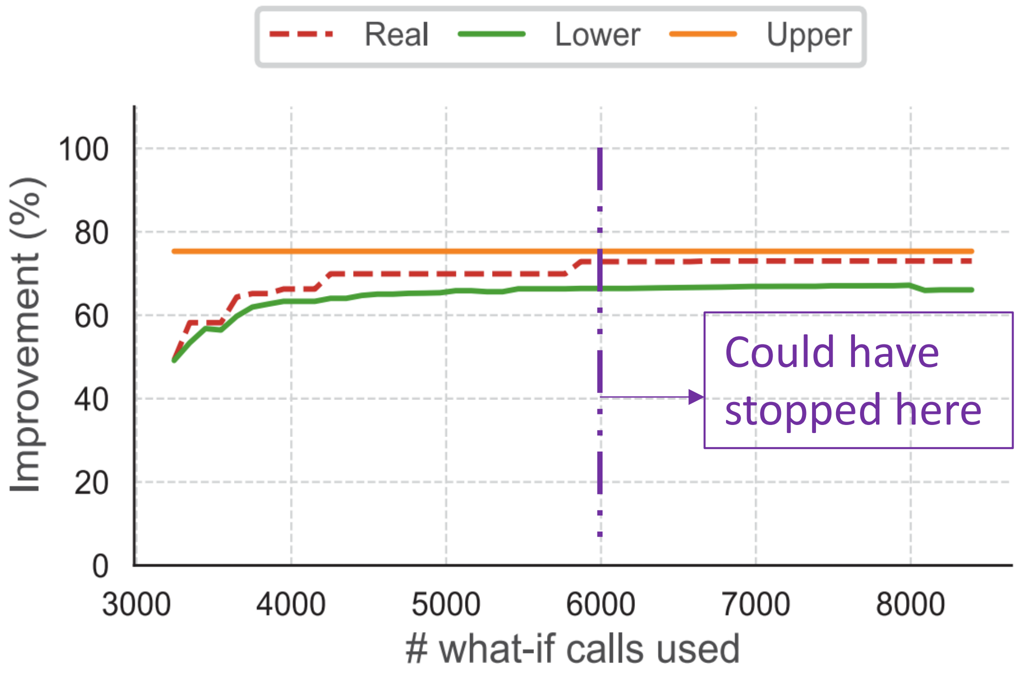


- What-if calls are expensive.
    - They dominate index tuning time.
    - **Example**: TPC-DS with 99 queries
- Budget-aware index tuning (SIGMOD 2022)
    - Constrain the number of what-if calls in configuration enumeration.
    - Find the best index configuration under the budget constraint.
    - Proposed algorithms: two-phase greedy (TPG), Monte Carlo tree search (MCTS)

Diminishing Return



(a) TPC-H, Two-phase greedy search



(b) Real-D, Monte Carlo tree search

Early Stopping for Budget-aware Index Tuning

- Define the *percentage improvement* of an index configuration C as
  - $\eta(W, C) = \frac{c(W,\emptyset) - c(W,C)}{c(W,\emptyset)} \times 100\% = \left(1 - \frac{c(W,C)}{c(W,\emptyset)}\right) \times 100\%$.
  - Here $\emptyset$ represents the existing index configuration.

- Let C_t^* be the *best* index configuration *after making t what-if calls*.
  - With this notation, C_B^* is the *best* index configuration after *using up all budget*.

- Stop making more what-if calls if $\eta(W, C_B^*) - \eta(W, C_t^*) < \epsilon$, where $0 \leq \epsilon \leq 1$ is some user-specified threshold.

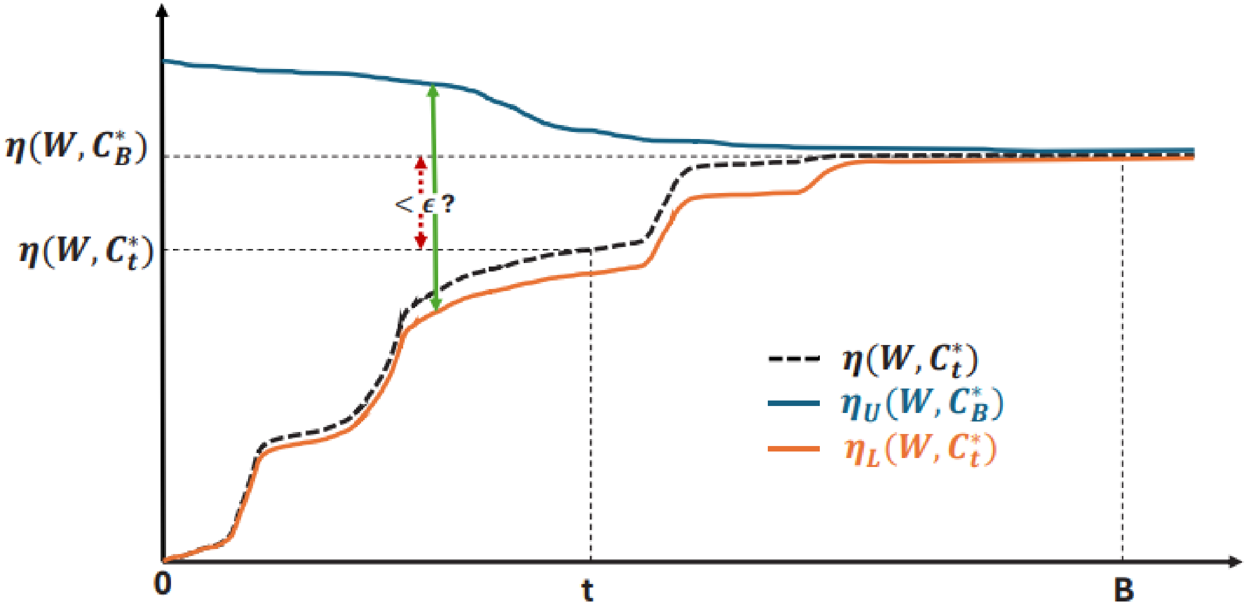
Challenges

- $\eta(W, C_B^*)$ is *unknown* with only t what-if calls made.

- $\eta(W, C_t^*)$ is *unknown* unless we make extra what-if calls.
  - As a common practice in budget-aware index tuning, we only know the *derived cost* $d(W, C_t^*)$ rather than the true what-if cost $c(W, C_t^*)$.
  - $d(W, C) = \sum_{q \in W} d(q, C)$, where $d(q, C) = min_{S \subseteq C} c(q, S)$.
  - Derived cost is much cheaper than making a what-if call but can be inaccurate.

Solutions

- Develop an upper-bound $\eta_U(W, C_B^*)$ of $\eta(W, C_B^*)$.
  - It is equivalent to developing a lower-bound $L(W, C)$ of $c(W, C)$.
- Develop a lower-bound $\eta_L(W, C_t^*)$ of $\eta(W, C_t^*)$.
  - It is equivalent to developing an upper-bound $U(W, C)$ of $c(W, C)$.
- Check if $\eta_U(W, C_B^*) - \eta(W, C_t^*) < \epsilon$.
  - This implies $\eta(W, C_B^*) - \eta(W, C_t^*) < \epsilon$.



Summary of Technical Details

- We extend the query-level lower bound $L(q, C)$ and upper bound $U(q, C)$ of $c(q, C)$, developed in our previous work, to the workload level.
  - Wii: Dynamic Budget Reallocation in Index Tuning, SIGMOD 2024.

- We propose both general versions of the workload-level lower/upper bounds and their optimized versions (i.e., *tighter bounds*) for greedy search.
  - Greedy search is used by both TPG and MCTS (as a building block).
  - We develop a "simulated greedy search" procedure to compute the optimized versions of the lower and upper bounds in a uniform manner.

- We further propose refined bounds by considering index interactions at workload level (e.g., benefits of similar indexes will be discounted).

Early-stopping Verification (ESV)

- There is computation overhead of ESV, i.e., computing the lower and upper bounds (e.g., by running simulated greedy search).

- As a result, we should not invoke ESV too frequently, which may result in nontrivial computation overhead.

- **Question:** When should we then invoke ESV?

Early-stopping Verification (ESV) Scheme

- A simple ESV scheme with fixed step size:
  - Invoke ESV after making $\{B_1, B_2, \ldots, B_n\}$ what-if calls, where $B_j = B_{j-1} + s$ for $1 \leq j \leq n$ (assuming $B_0 = 0$) and s is the step size.
  - **Problem**: It can result in many unnecessary ESV invocations and thus significant computation overhead.

- We propose an ESV scheme by monitoring the convexity and concavity of the "index tuning curve" (ITC) to reduce overhead.

ESVS by Monitoring Convexity/Concavity of ITC

Start with the fixed step-size ESV scheme: $B_j = B_{j-1} + s$

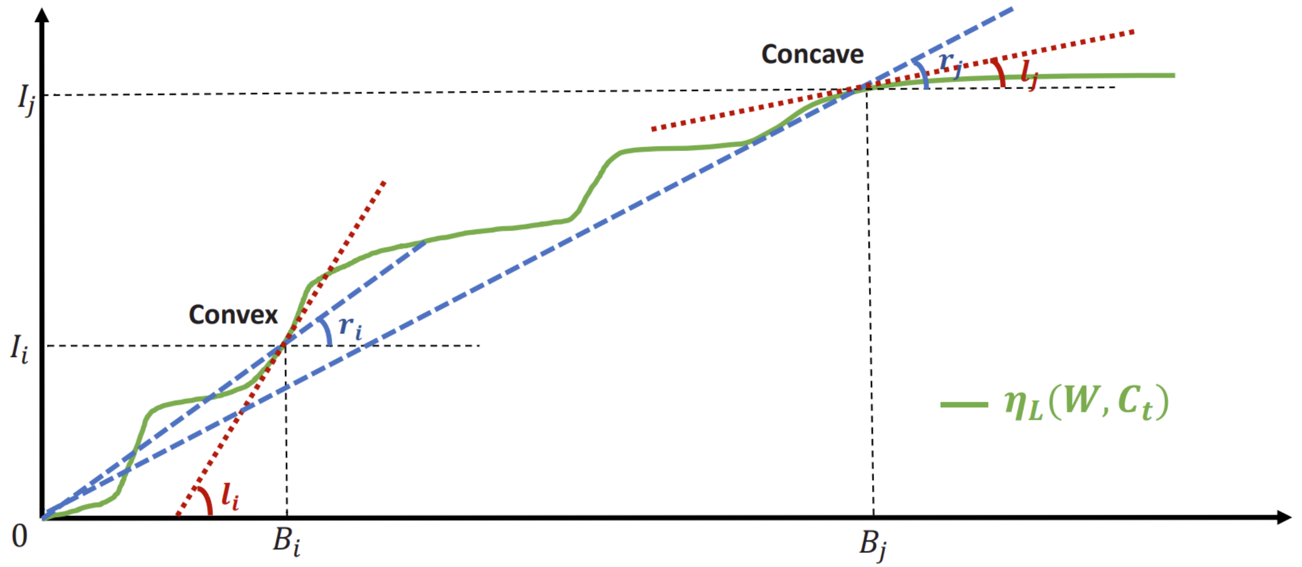
Check the "<u>significance of concavity</u>" σ_{j+1} at B_{j+1}:

$$\sigma_{j+1} = \frac{\delta_{j+1}}{\Delta_{j+1}} = \frac{p_{j+1}^r - I_{j+1}}{p_{j+1}^r - p_{j+1}^l}$$

$$p_{j+1}^r = I_j + r_j \cdot (B_{j+1} - B_j)$$

$$p_{j+1}^l = I_j + l_j \cdot (B_{j+1} - B_j)$$

Invoke ESV if $\sigma_{j+1} \geq \sigma$, where $0 < \sigma < 1$ is some threshold.



The "index tuning curve" (ITC) represents a function from the number of what-if calls made to the improvement observed.

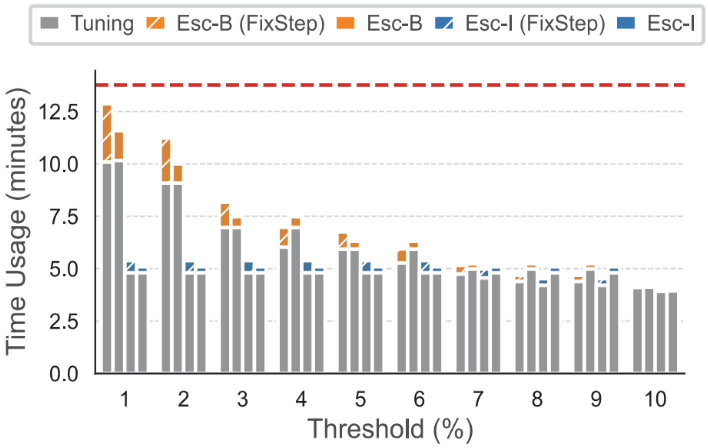
<u>Improvement Rate</u>: $r_j = \frac{I_j - I_0}{B_j - B_0}$

<u>Latest Improvement Rate</u>: $l_j = \frac{I_j - I_{j-1}}{B_j - B_{j-1}}$

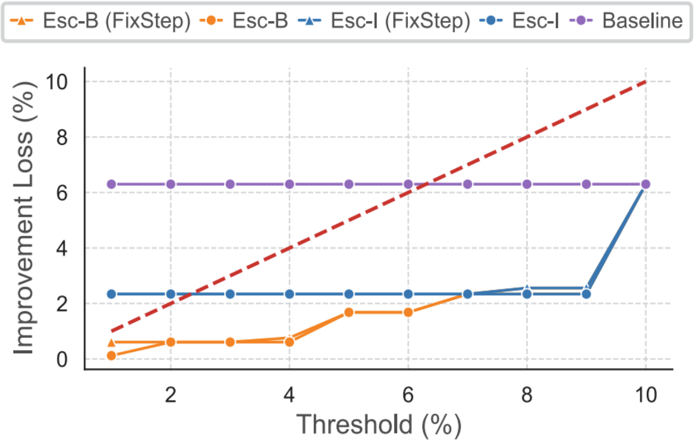
<u>Convex:</u> $l_j > r_j$; **<u>Concave:</u>** $l_j < r_j$.

Experimental Evaluation Results (TPC-H)

Two-phase greedy search (K = 20, B = 20k)

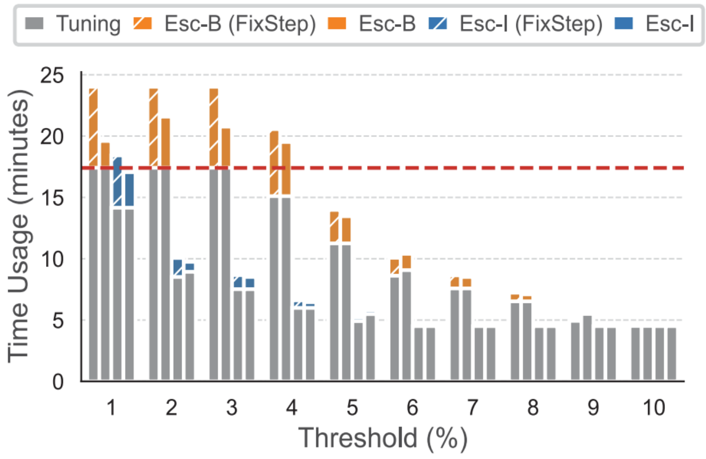


(a) Time Overhead

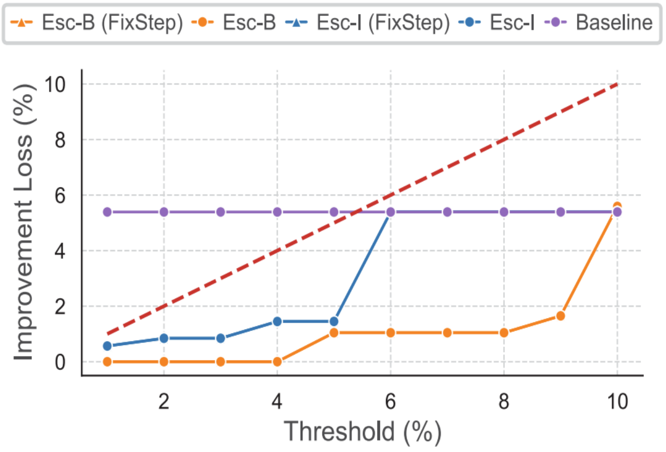


(b) Improvement Loss

Monte Carlo tree search (K = 20, B = 20k)



(a) Time Overhead



(b) Improvement Loss

K is the # of indexes allowed.

B is the budget on the number of what-if calls.

Improvement loss: The actual improvement loss (not the threshold ϵ) when exiting index tuning w/o making all what-if calls.

Takeaways

- We proposed and formulated the problem of "early stopping" in budget-aware index tuning.
- We proposed lower and upper bounds of the improvement given by the best configuration after making a certain number of what-if calls to check the early-stopping condition.
- We proposed monitoring the convexity and concavity of the index tuning curve to reduce the number of ESVs and thus reduce the overall computation overhead of invoking ESVs.

Backup Slides

Query-level Lower/Upper Bounds (Wii, SIGMOD 2024)

Assumption 1 (**Monotonicity**): Let C_1 and C_2 be two index configurations where $C_1 \subseteq C_2$. Then $c(q, C_2) \leq c(q, C_1)$.

Assumption 2 (**Submodularity**): Given two configurations $X \subseteq Y$ and an index $z \notin Y$, we have $c(q, Y) - c(q, Y \cup \{z\}) \leq c(q, X) - c(X \cup \{z\})$.

- Upper bound is set as the derived cost $U(q, C) = d(q, C) = min_{S \subseteq C} c(q, S)$.
- Lower bound is set as $L(q, C) = c(q, \emptyset) - \sum_{z \in C} u(q, z)$
  - $u(q, z)$ is the *upper bound* of the marginal cost improvement (MCI) of z.
  - Example 1: $u(q, z) = c(q, \emptyset) - c(q, \{z\}) = \Delta(q, \{z\})$
  - Example 2: $u(q, z) = c(q, \emptyset) - c\left(q, \Omega_{\mathrm{q}}\right) = \Delta(q, \Omega_q)$

Ω_{q} represents the "best possible configuration" with **ALL** candidate indexes of q included.