Acknowledgments

The research project is implemented in the framework of H.F.R.I call “3rd Call for H.F.R.I.” Research Projects to Support Faculty Members & Researchers” (H.F.R.I. Project Number: 25146).

Adaptive Indexing

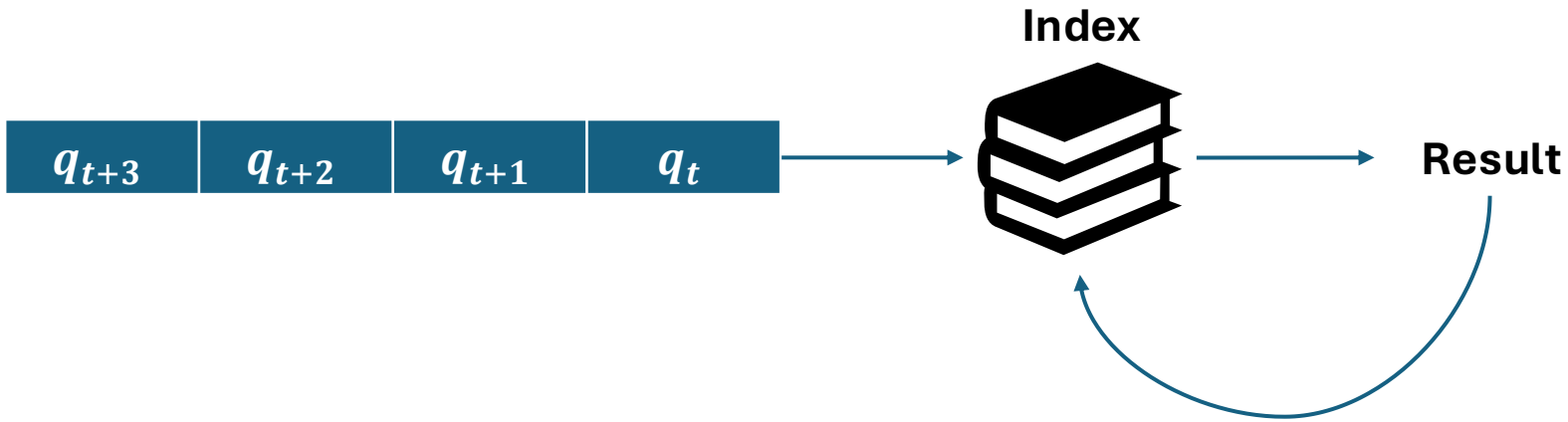
A short presentation of relative papers

1. CRACK-IVF
2. Quake

Dimitris Petratos

The Problem

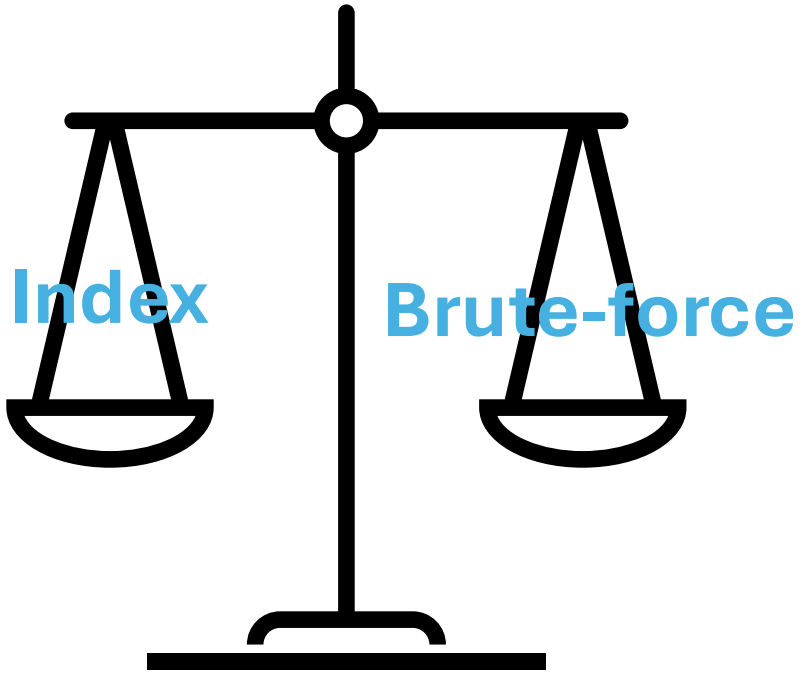
- Update/adapt the index, as new queries arrive



CRACK-IVF ~ The need for adaptive indexing

Construction Cost (Time+Memory)

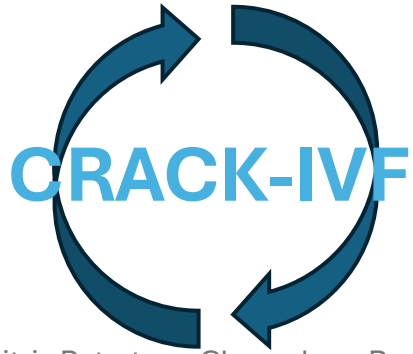
Search Time (Good Scaling)



Search Time (Bad Scaling)

No Construction Cost

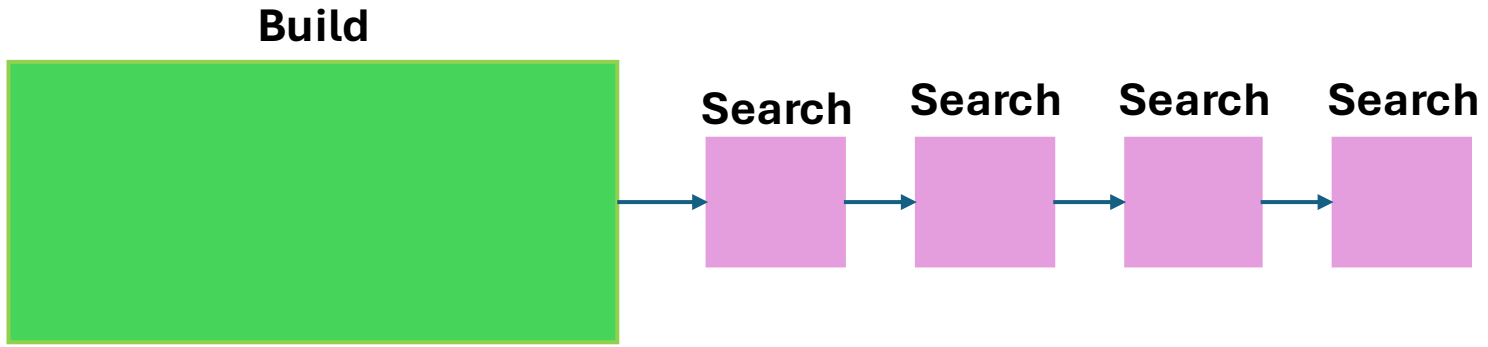
Vasilis Mageirakos, Bowen Wu, and Gustavo Alonso. 2025. Cracking Vector Search Indexes. Proc. VLDB Endow. 18, 11 (July 2025), 3951–3964. https://doi.org/10.14778/3749646.3749666



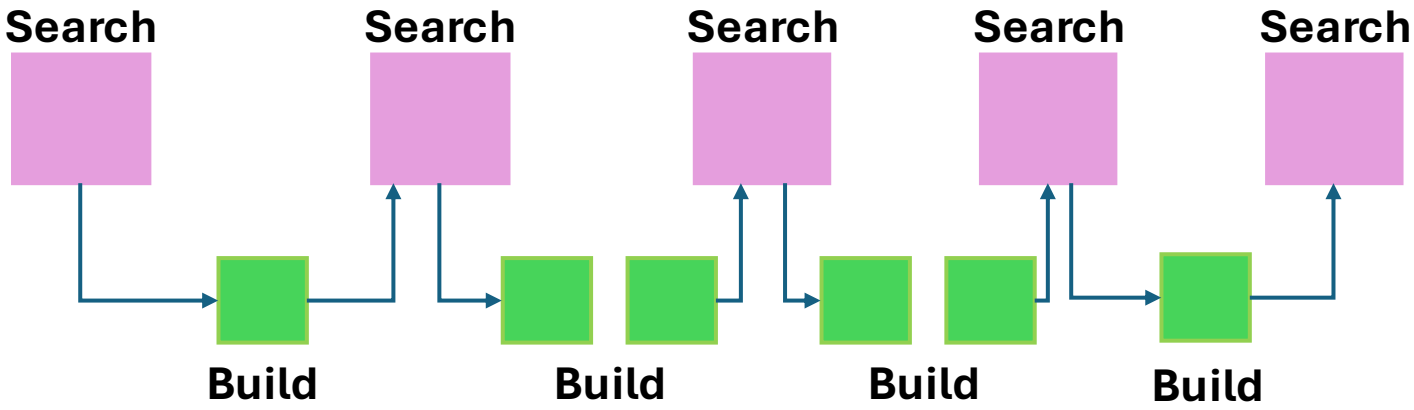
Build the index as you query

CRACK-IVF ~ The paradigms

Static Paradigm



Adaptive Paradigm (CRACK-IVF)



CRACK-IVF ~ The operations

SEARCH

Search for the k-NN of a query point. Returns the nearest neighbors, their distances and the local partition IDs visited.

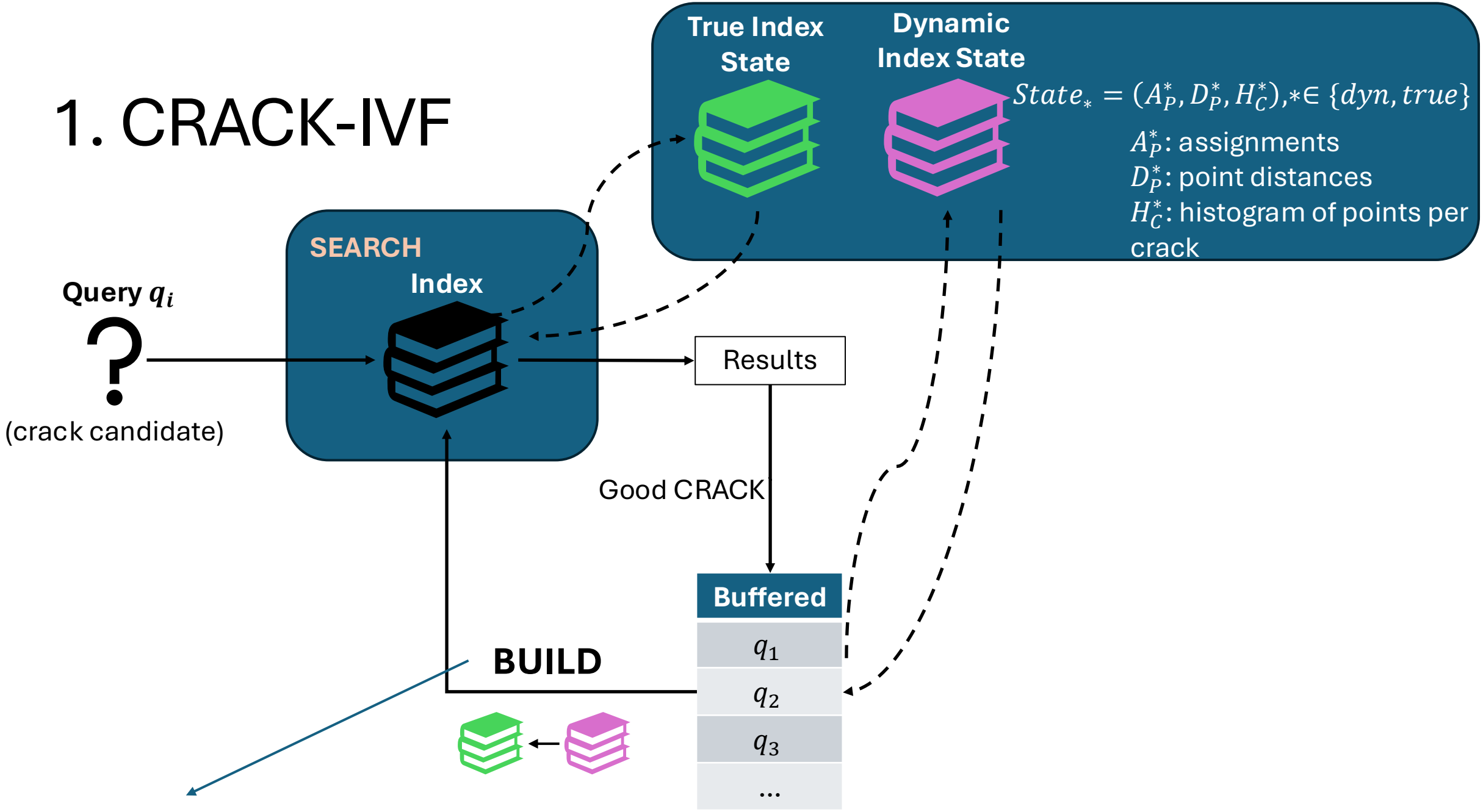
CRACK

Introduces entirely new partitions to the index. Occurs in frequently accessed regions that require finer partitioning.

REFINE

Improves current partitions using localized K-Means. Adjusts centroids to distribute points more evenly in a local region.

1. CRACK-IVF

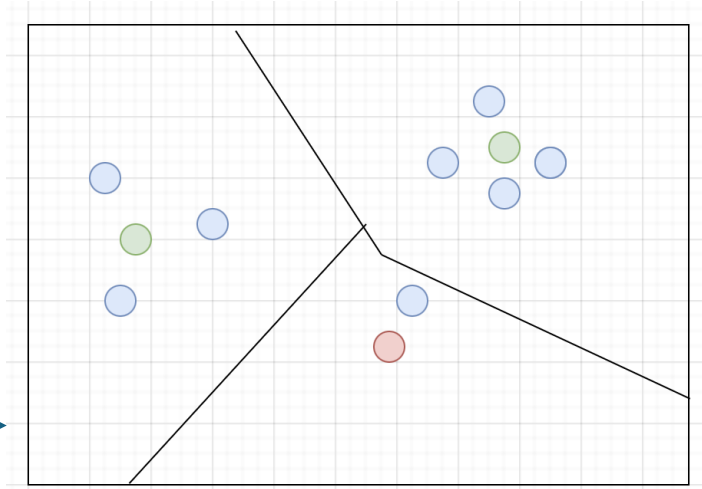

True Index State
Dynamic Index State
$State_* = (A_P^*, D_P^*, H_C^*), * \in \{dyn, true\}$
A_P^*: assignments
D_P^*: point distances
H_C^*: histogram of points per crack
SEARCH
Index
Query q_i
(crack candidate)
Results
Good CRACK
Buffered
q_1
q_2
q_3
...
BUILD
BUILD∈{CRACK, REFINE}

CRACK-IVF ~ WHERE to Build

Rule 1: WHERE to CRACK

Every query is a candidate, unless it fails rejection criteria:

- $|I_{steal}| < min_pts\ (default = 2)$
- $\frac{Total\ local\ points}{Total\ local\ cracks} > pts_crack_thr\ (default = 64)$



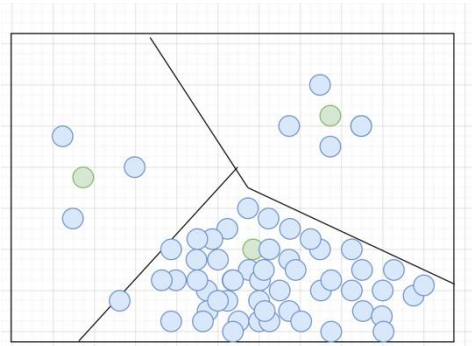
Bad candidate

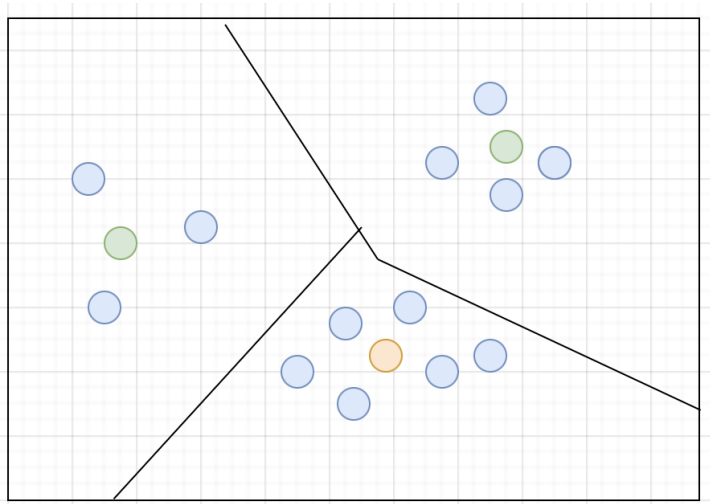
Existing cracks

Rule 2: WHERE to REFINE

Trigger REFINE iff:

- $CV_{local} = \frac{\sigma_{local}}{\mu_{local}} > cv_\max\ (default = 2)$
- $\exists c_1, c_2 : S(c_1) \leq cutoff_{low}, S(c_2) \geq cutoff_{high}$





Good candidate (Buffered crack)

Existing cracks

1. CRACK-IVF ~ Building Heuristics WHEN to BUILD

Kernel	Compute	Data Movement
Get Local Region†	$\lvert C_{\text{buffered}}\rvert \times \lvert C\rvert$	$\lvert P_{\text{local}}\rvert \times d$
Commit Reorg (w/o A_P^{dyn})	$\lvert P_{\text{local}}\rvert \times \lvert C_{\text{local}}\rvert \times d + \lvert C\rvert$	$\lvert P_{\text{local}}\rvert \times d$
Commit Reorg (w/ A_P^{dyn})	$\lvert C\rvert$	$\lvert P_{\text{local}}\rvert \times d$
Update Centroids	$\lvert P\rvert \times d$	$(\lvert P\rvert + \lvert C\rvert) \times d$
Local K-Means	$n_{\text{iter}} \times \lvert P_{train}\rvert \times \lvert C_{\text{local}}\rvert \times d$	$\lvert C\rvert \times d$

- T_{build}: total measured time spent on previous build operations
- T_{search}: cumulative time spent on all past search operations
- $\hat{T}_f, f \in \{CRACK, REFINE\}$: estimated execution cost of operation f (a predictive cost model is used)
- Budget constraint:

$$T_{build} + \hat{T}_f \leq a \cdot (T_{build} + T_{search} + \hat{T}_f) \ \ (a = 0.5)$$

- $T_f = \sum_i T_{f,i}, \forall f \in \{CRACK, REFINE\} \rightarrow T_{f,i} = w_{f,i}^1 \cdot C_{f,i} + w_{f,i}^2 \cdot D_{f,i} + b_{f,i}$

Learnable coefficients

$i \in \{Get\ Local\ Region, Commit\ Reorg(w|A_P^{dyn}), Update\ Centroids, Local\ K-Means, Commit\ Reorg(w|o\ A_P^{dyn})\}$

Dominant computational complexity of i

Dominant data movement cost of i

2. Quake

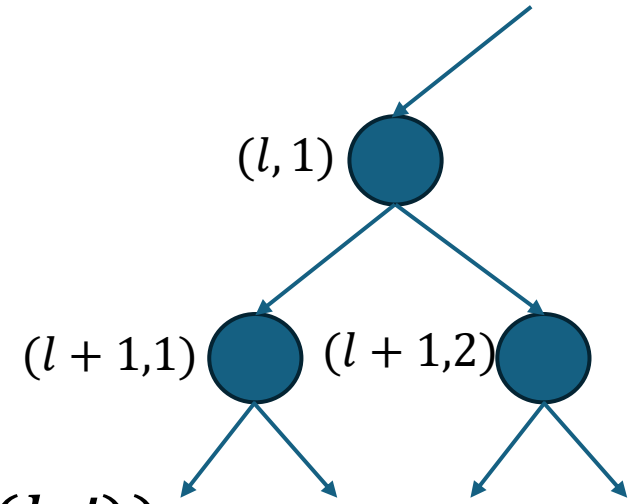
- Adaptive Incremental Maintenance:
  - ➢Split Partition→K-Means to split a partition into two partitions
  - ➢Merge Partition→Remove a partition and reassign its points to other partitions
  - ➢Add Level→Add a level of partitioning
  - ➢Remove Level→Remove current top level and merge next level's partitions
- Adaptive Partition Scanning (APS):
- Numa-aware query processing

Jason Mohoney, Devesh Sarda, Mengze Tang, Shihabur Rahman Chowdhury, Anil Pacaci, Ihab F. Ilyas, Theodoros Rekatsinas, and Shivaram Venkataraman. 2025. Quake: adaptive indexing for vector search. In Proceedings of the 19th USENIX Conference on Operating Systems Design and Implementation (OSDI '25). USENIX Association, USA, Article 9, 153–169.

2. Quake - Adaptive Incremental Maintenance

Cost Model

$$C = \sum_{l=0}^{L-1} \sum_{j=0}^{N_l-1} A_{l,j} \lambda(s_{l,j})$$



- (l, j): partition j at level l
- $A_{l,j} \in [0,1]$: access frequency (#queries scanned in (l, j))
- $\lambda(s)$: latency function for scanning s vectors→offline profiling

2. Quake - Adaptive Incremental Maintenance

Cost Model's Interpretation

- Larger partitions→more time to scan→increasing latency/reduced number of partitions needed
- Smaller partitions→reduced number of vectors needed to be scanned to reach high recall

Guiding Maintenance Decisions

$$\Delta \mathrm{C} = \mathrm{C}_{\mathrm{after}} - \mathrm{C}_{\mathrm{before}}$$

Actions (splitting/merging etc) are applied only if $\Delta C \leq -\tau$

2. Quake - Adaptive Incremental Maintenance

Split Partition

When a partition is too large or frequently accessed:

- Conduct K-Means(2) and create partitions (l, j_L) and (l, j_R)
- Refine partitions

Merge Partition

When a partition is rarely accessed

- Delete the partition
- Reassign its points to their respective nearest partitions

Add/Remove Levels

If top level's centroids are dense→cluster its centroids

If top level's centroids are sparse→remove top level and merge its centroids

$$\Delta O^+ = \lambda(N_l + 1) - \lambda(N_l)$$
$$\Delta O^- = \lambda(N_l - 1) - \lambda(N_l)$$

2. Quake – Actions' Cost

- Exact Split Delta:

$$\Delta Split_{l,j} = \Delta O^+ - A_{l,j}\lambda(s_{l,j}) + A_{l,j_L}\lambda(s_{lj_L}) + A_{l,j_R}\lambda(s_{lj_R})$$

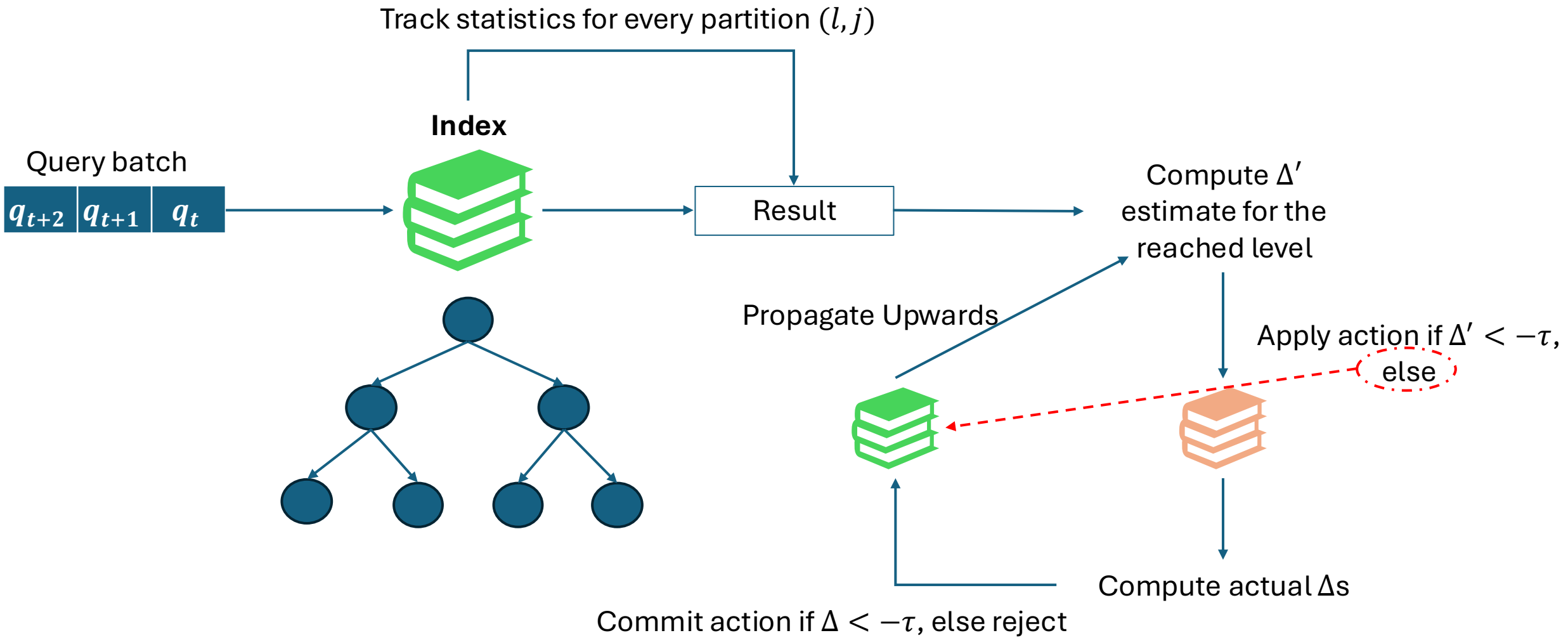
- Approximate Split Delta:

$$\Delta' Split_{l,j} = \Delta O^+ - A_{l,j}\lambda(s_{l,j}) + 2\alpha A_{l,j}\lambda\left(\frac{s_{lj}}{2}\right)$$

- Exact Merge Delta:

$$\Delta Merge_{l,j} = \Delta O^- - A_{l,j}\lambda(s_{l,j}) + \sum_{m \in R_{l,j}}^{b} [(A_m + \Delta A_m)\lambda(s_m + \Delta s_m) - A_m\lambda(s_m)]$$

2. Quake - Workflow



2. Quake – Adaptive Partitioning Scanning (APS)

- Dynamically determines the number of partitions to scan per query, to achieve a specified recall target
- Probability that partition P_i contains one of the k-nns of query q

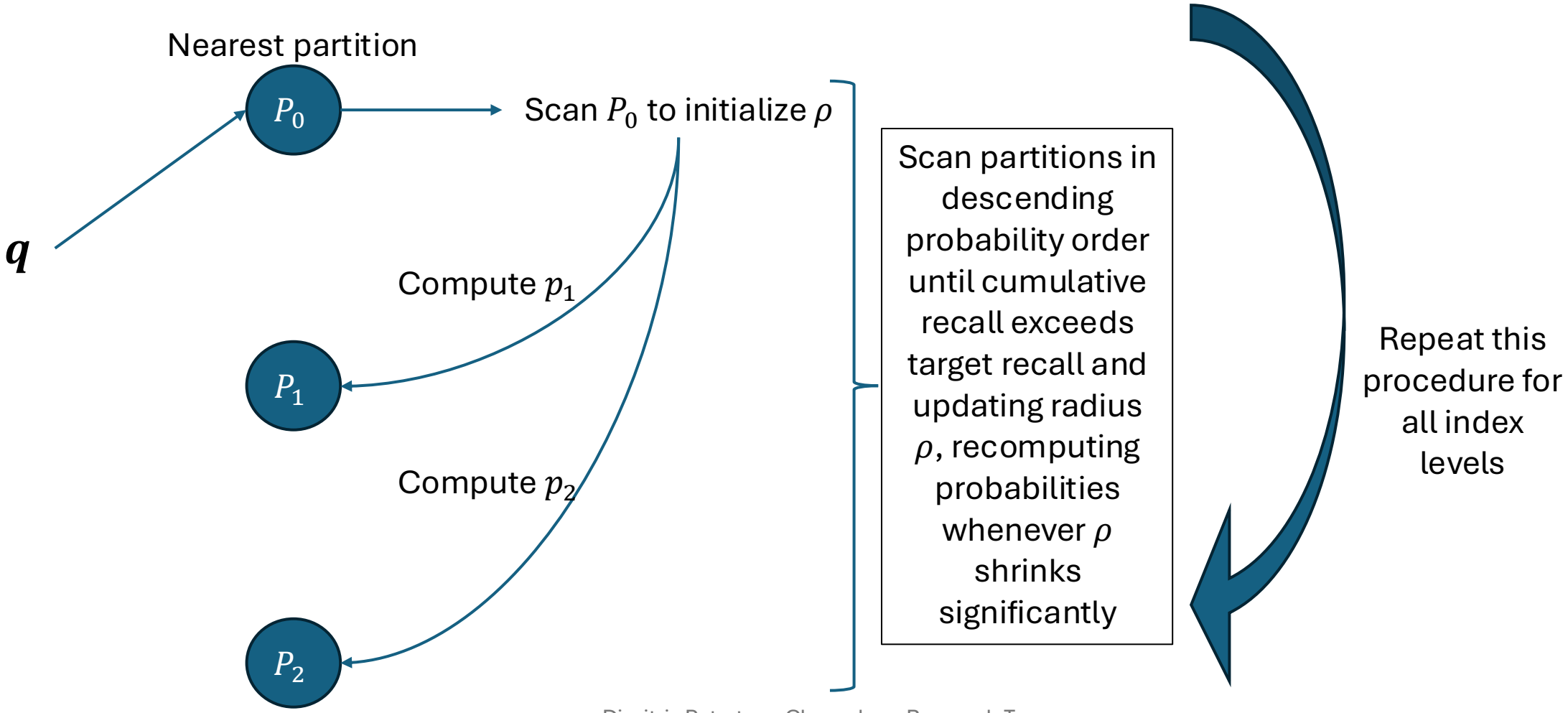
$$p_i = \frac{Vol(B(q,\rho) \cap P_i)}{Vol(B(q,\rho))}$$

where ρ is the distance of q to its k-nn.

ρ is estimated as the current k-nn distance observed and it is updated online as partitions are scanned.

$Vol(B(q,\rho) \cap P_i)$ is approximated

2. Quake – Adaptive Partitioning Scanning (APS)



Acknowledgments

The research project is implemented in the framework of H.F.R.I call “3rd Call for H.F.R.I.” Research Projects to Support Faculty Members & Researchers” (H.F.R.I. Project Number: 25146).