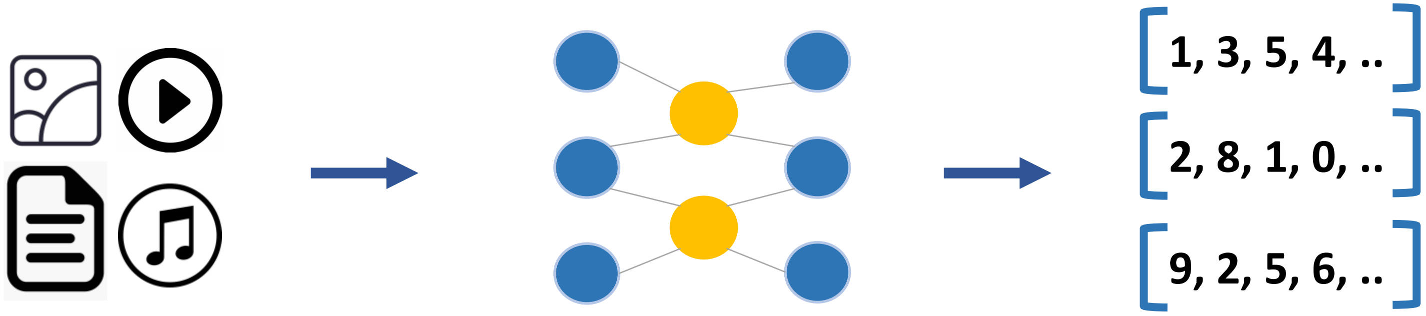
SPFresh: Incremental In-Place Update for Billion-Scale Vector Search

Yuming Xu[1] Hengyu Liang[1] Jin Li[3] Shuotao Xu[2] Qi Chen[2] Qianxi Zhang[2]
Cheng Li[1] Ziyue Yang[2] Fan Yang[2] Peng Cheng[2] Mao Yang[2]

[1]USTC [2]Microsoft Research Asia [3]Harvard University

1 USTC, CHINA ADSLAB 2 Microsoft 3

Vectors: the key data type in AI era

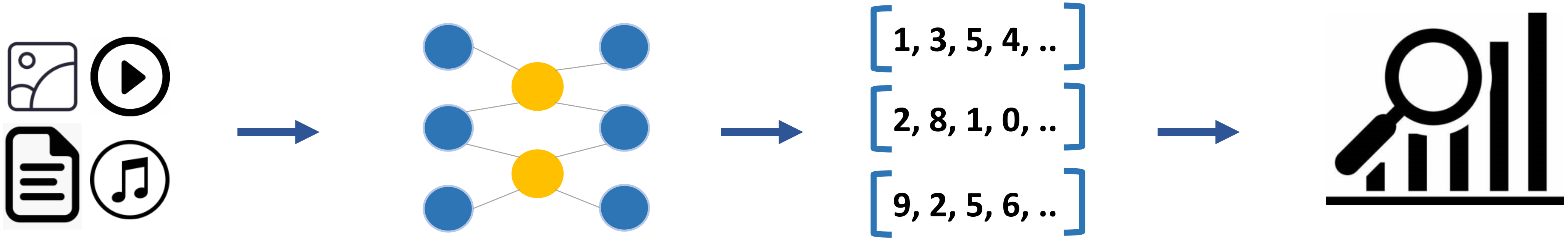


Raw Data — Deep learning model — High-dimensional vectors

E.g. images, videos, texts..

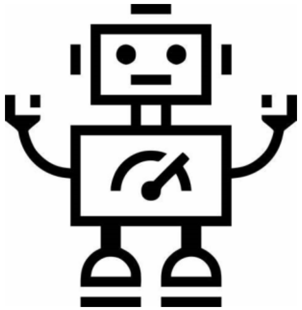
High-dimensional Space (distance between vectors represents raw data similarity)

Vectors: the key data type in AI era

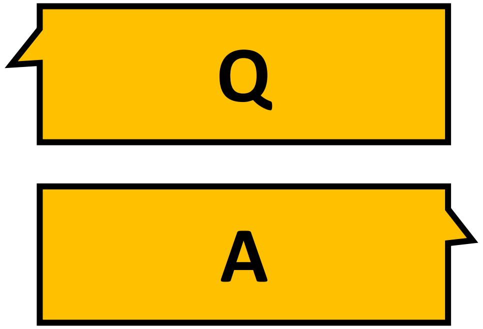


Raw Data | Deep learning model | High-dimensional vectors | **Vector similarity search**

Recommendation



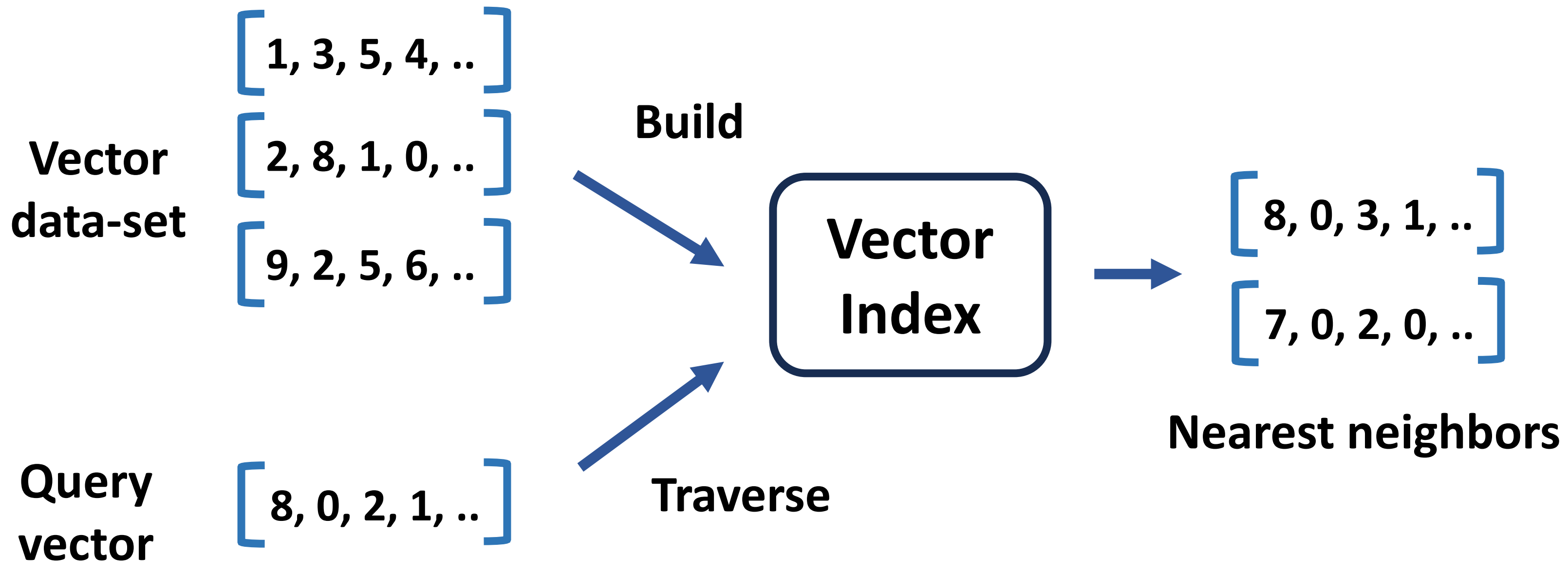
LLM retrieval plugin



Query answering

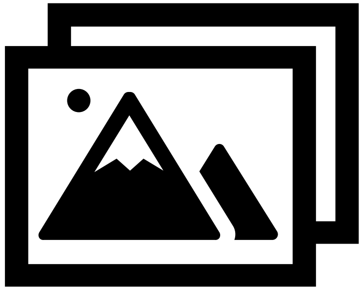
Vector similarity search empowers semantic-understanding tasks

Vector index: the key component for search



- **Vector index allows low-latency, qualitative approximate vector search**
  - Exact search in a high-dimensional space is unscalable
  - Trade in small search accuracy for much lower search latency
  - Works well for billion-scale data-set

Applications requires frequent update to index



500+ hours of content are uploaded to YouTube **every minute** [1]

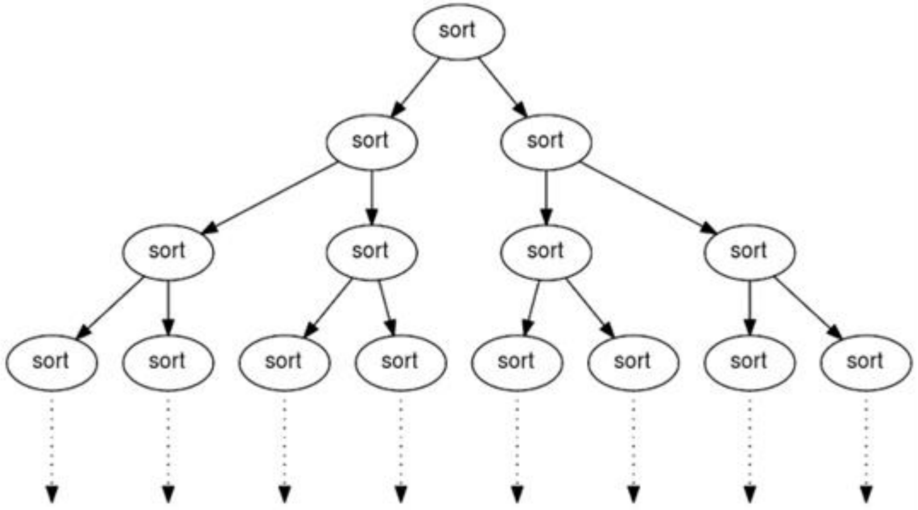
One billion new images are updated in JD.com **every day** [2]

500PB unstructured data are ingested to Alibaba **during a shopping festival** [3]

[1] Youtube. https://blog.youtube/press/
[2] Li et al. The Design and Implementation of a Real Time Visual Search System on JD E-Commerce Platform. (Middleware’18)
[3] Wei et al. AnalyticDB-V: A Hybrid Analytical Engine towards Query Fusion for Structured and Unstructured Data. (VLDB'20

Vector index: complex abstraction

- Proximity in high dimension is hard to organize
- Inefficient vector index affects **the query accuracy**



Scalar index

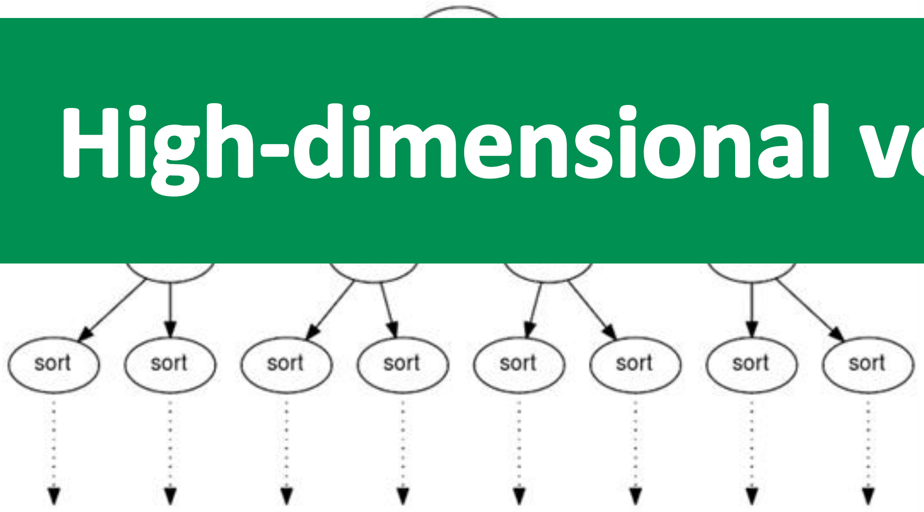
Based on scalar value order

Vector index

Based on proximity in a high dimensional space

Vector index: complex abstraction

- Proximity in high dimension is hard to organize
- Inefficient vector index affects **the query accuracy**



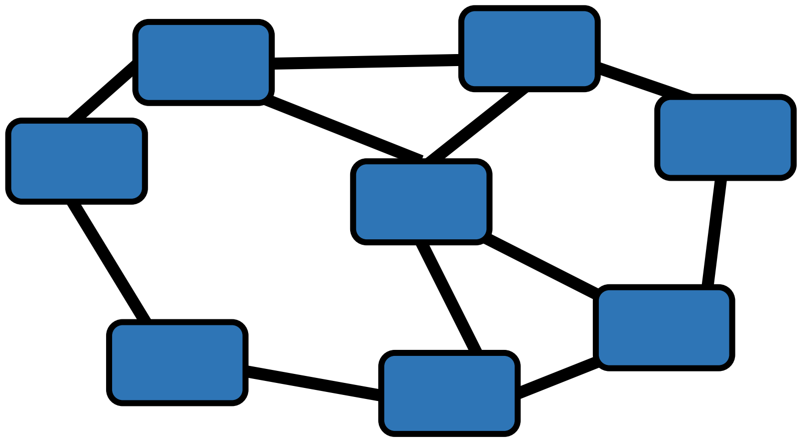
High-dimensional vector index is hard to update

Scalar index
Based on scalar value order

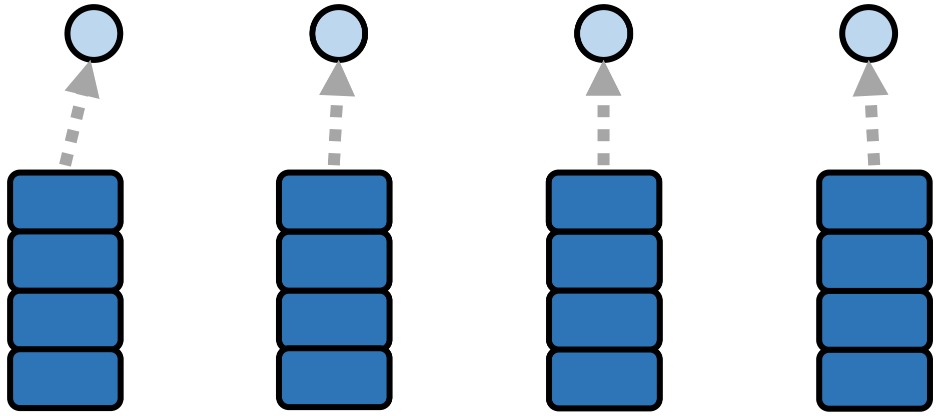
Vector index
Based on proximity in a high dimensional space

Common vector index organizations

- For billion-scale vector scenario, vector index can be categorized into



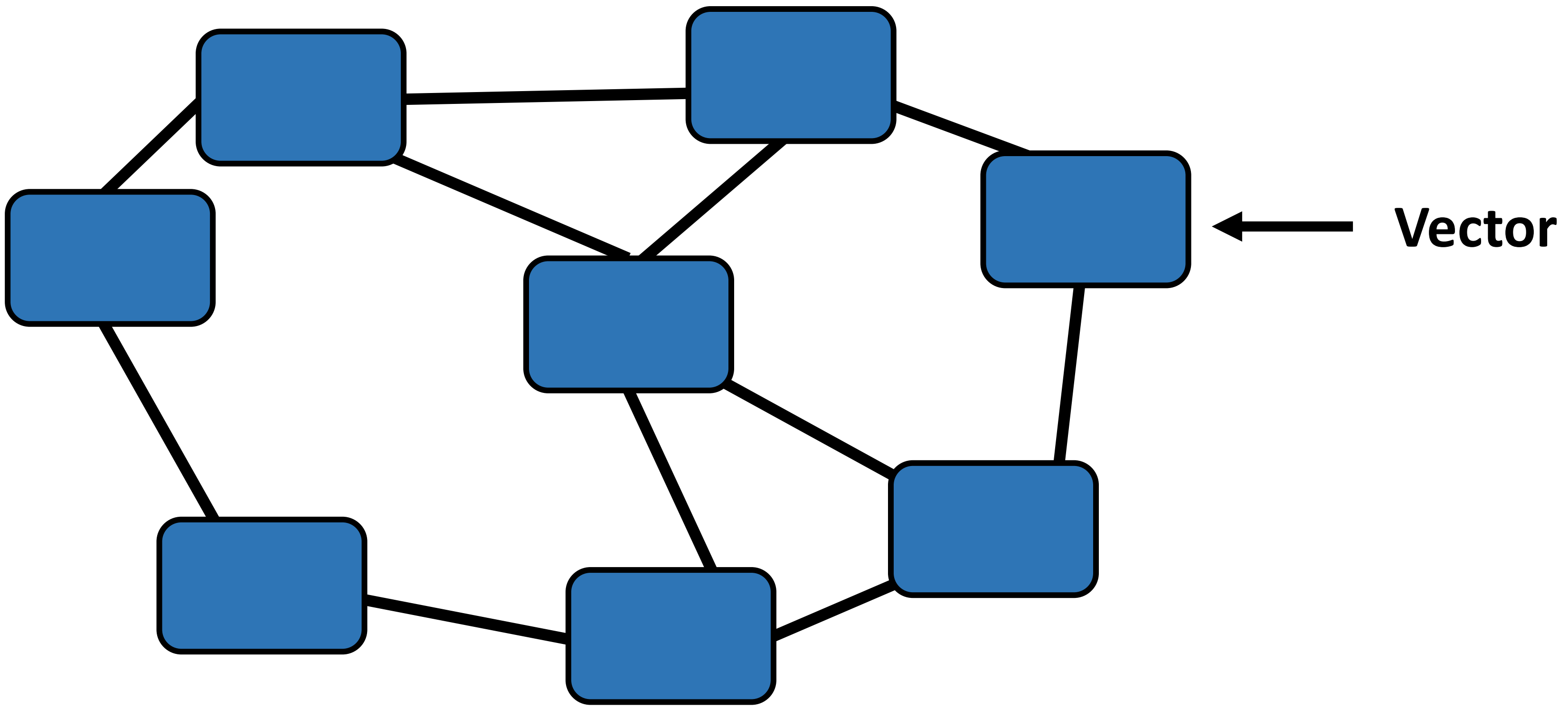
Fine-grained graph vector index



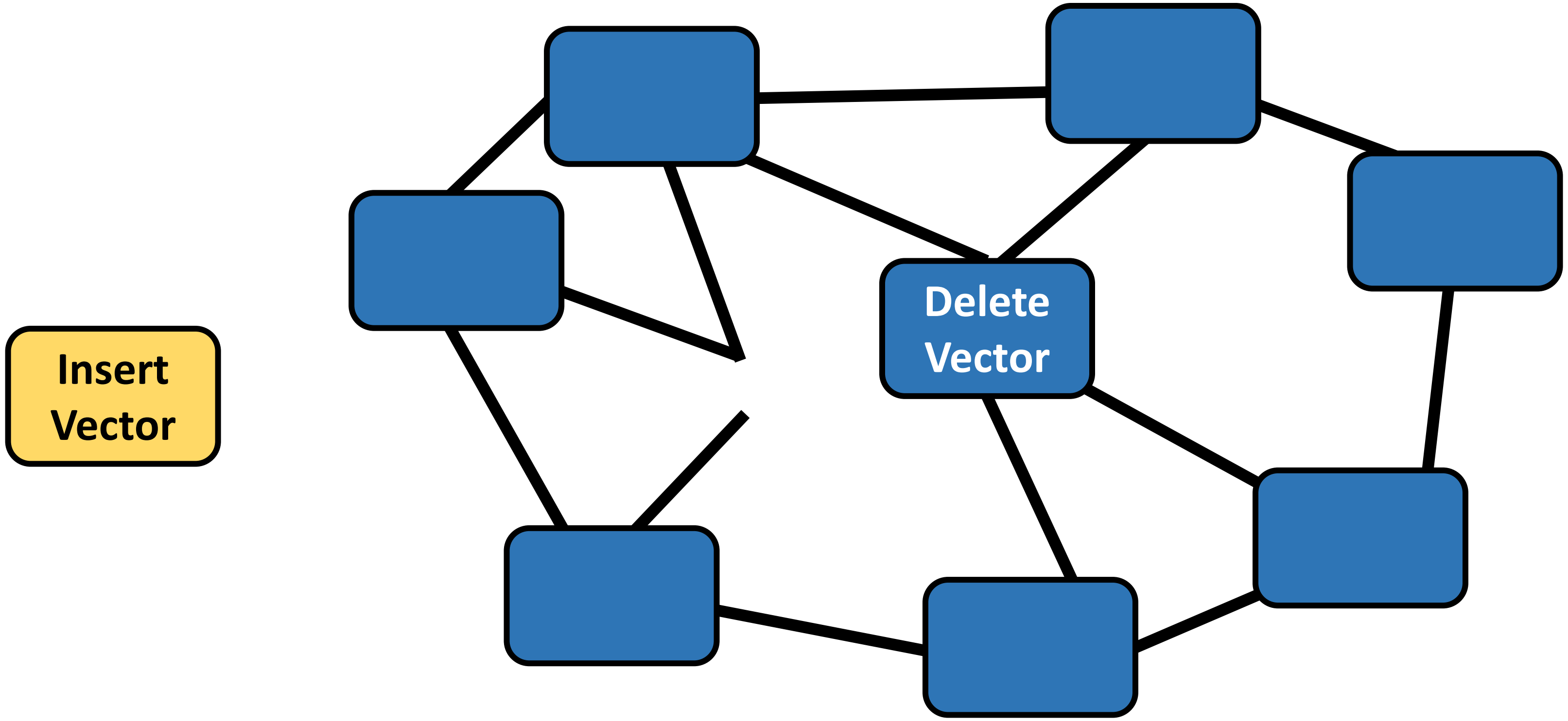
Coarse-grained cluster vector index

Common vector index organizations

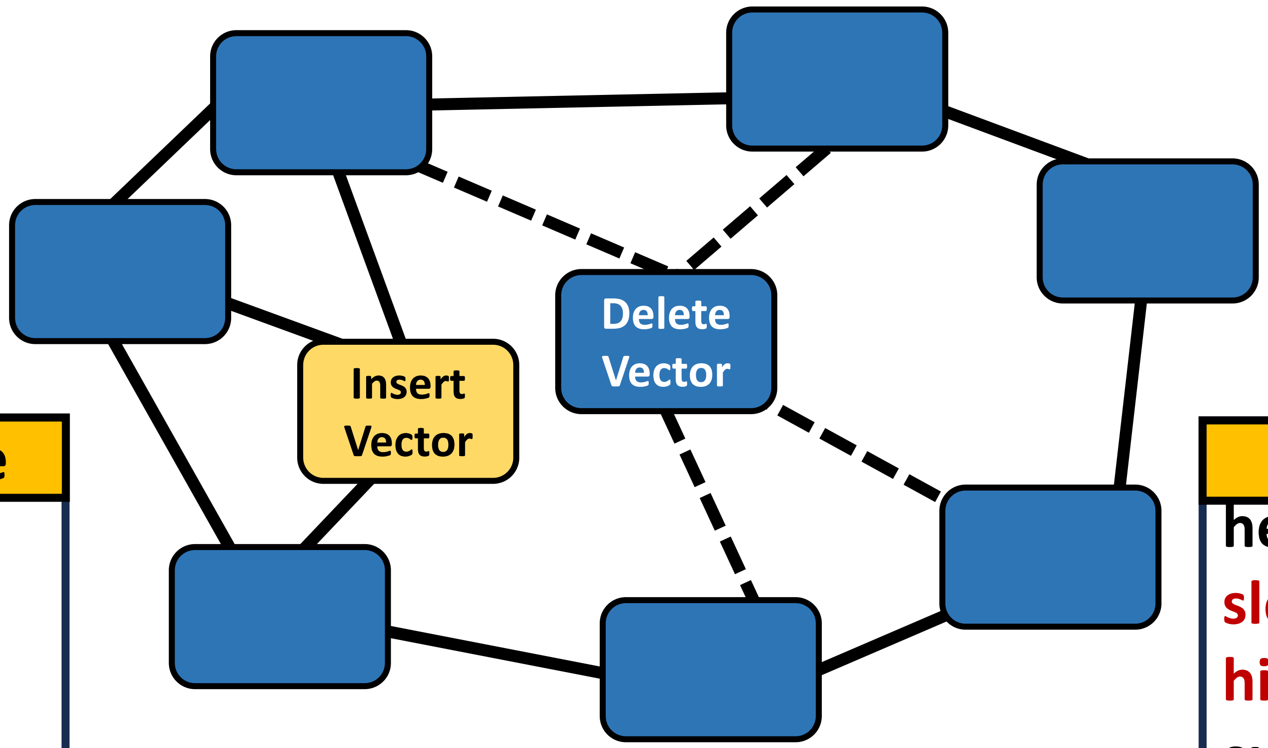
- Fine-grained graph-based vector indices
  - Connect vectors with short distance



Updating fine-grained graph is challenging!



Updating fine-grained graph is challenging!



Insufficient update

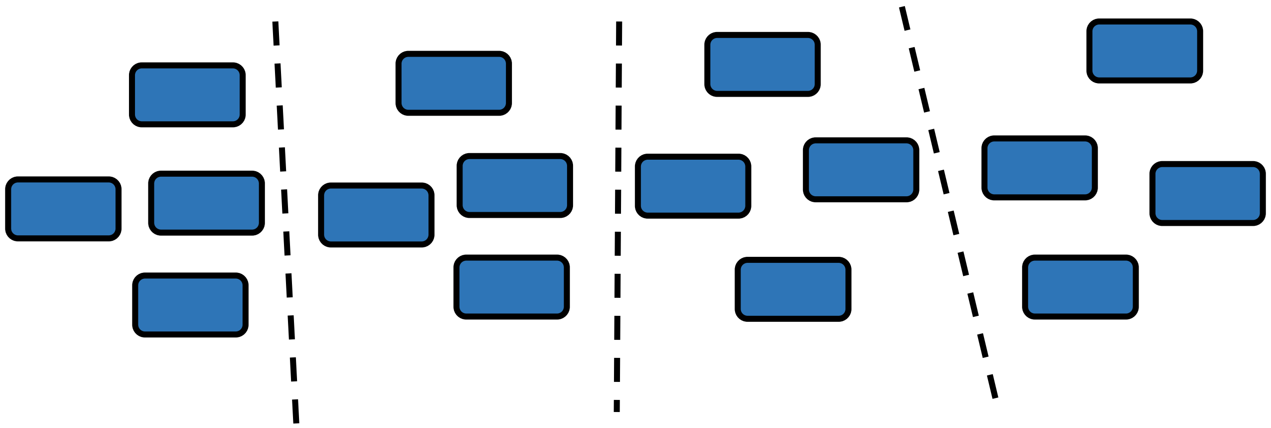
light scan,
fast update,
More to search,
Dropped accuracy

Sufficient update

heavy scan,
slow update,
high resource usage,
sustainable accuracy

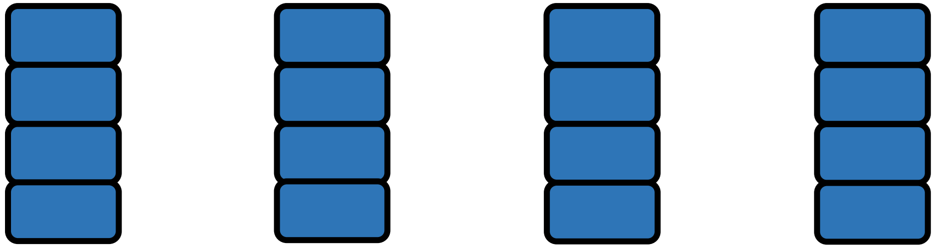
Common vector index organizations

- Coarse-grained cluster-based vector indices
  - Collect vectors in close proximity into the same partition



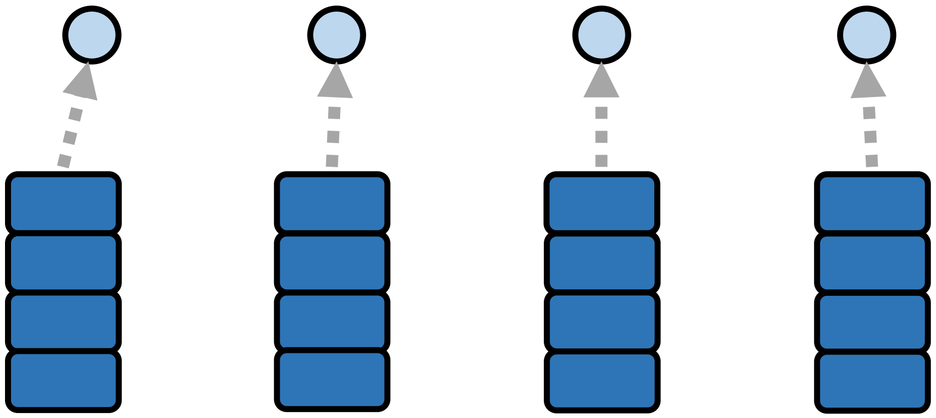
Common vector index organizations

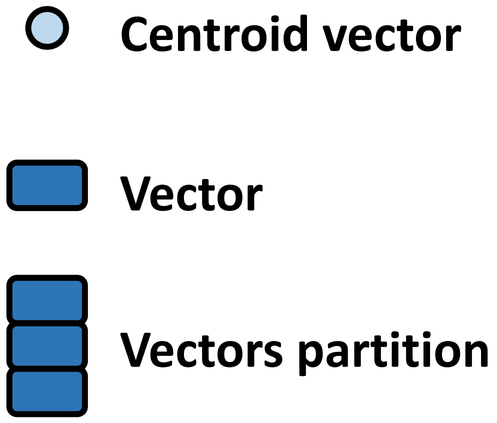
- Coarse-grained cluster-based vector indices
  - Collect vectors in close proximity into the same partition



Common vector index organizations

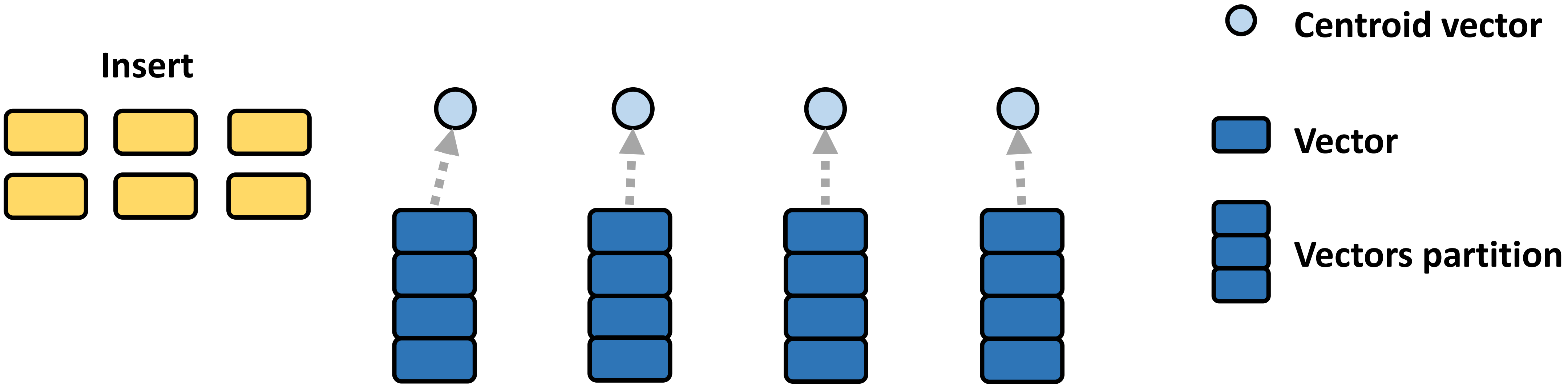
- Coarse-grained cluster-based vector indices
  - Collect vectors in close proximity into the same partition
  - The centroid of partition **represents vectors in the partition**





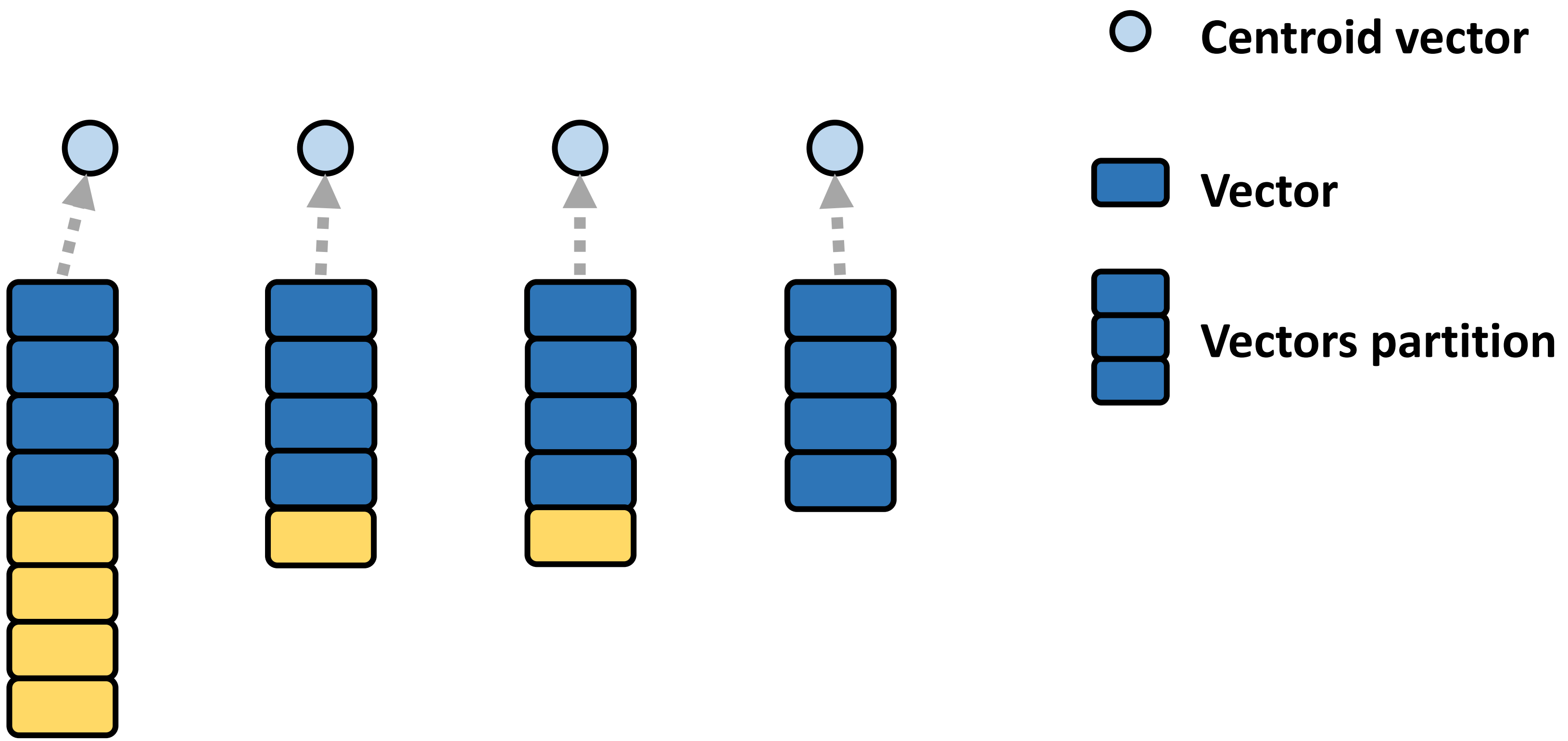
Coarse-grained cluster is cheap to update, but...

- Inserts/deletes only modify the corresponding partitions



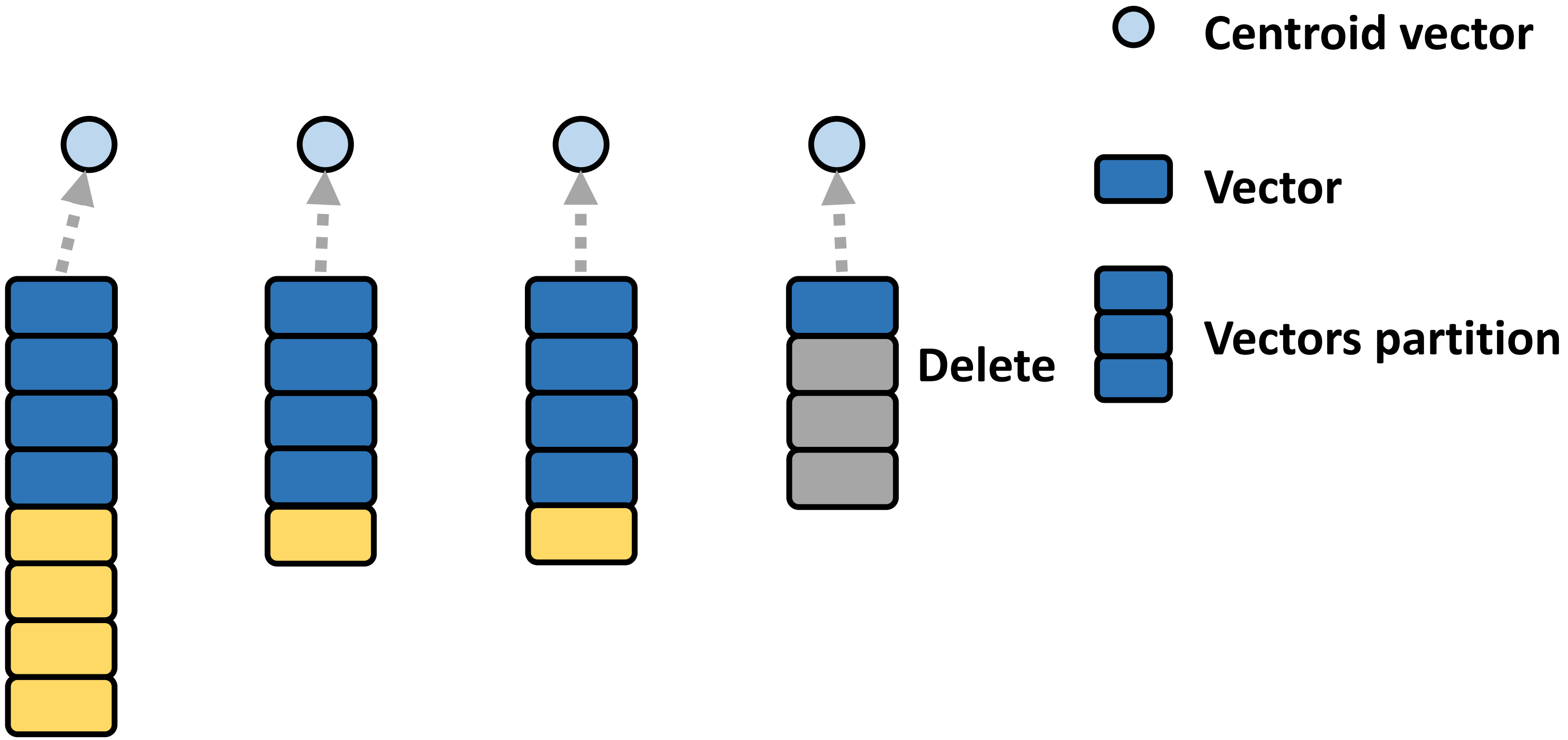
Coarse-grained cluster is cheap to update, but...

- Inserts/deletes only modify the corresponding partitions



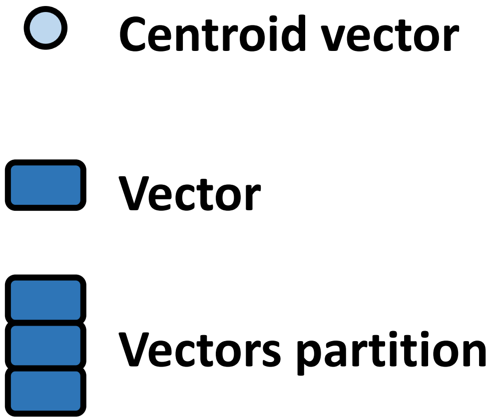
Coarse-grained cluster is cheap to update, but...

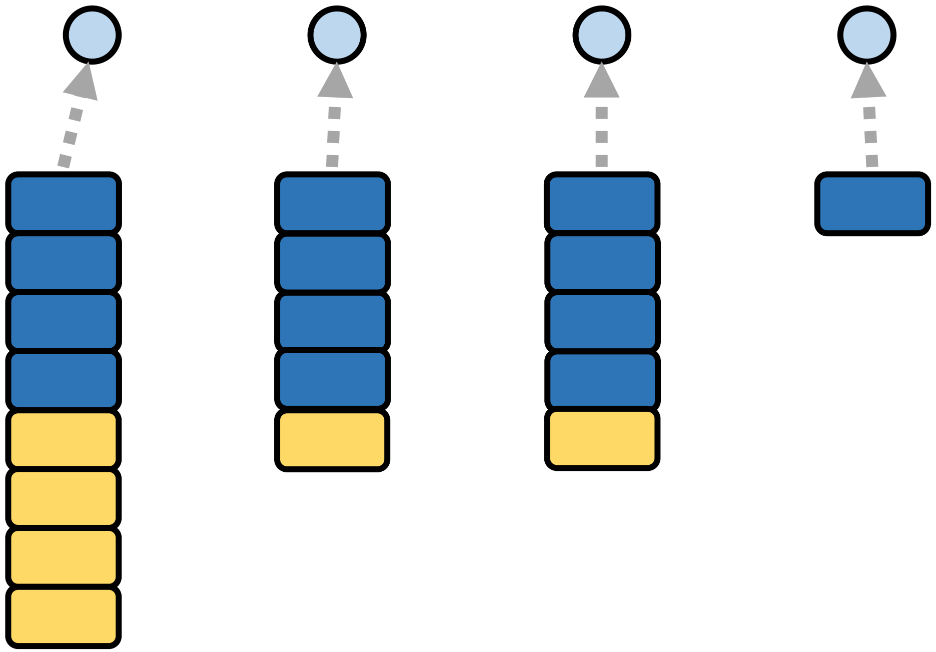
- Inserts/deletes only modify the corresponding partitions



Coarse-grained cluster is cheap to update, but...

- Inserts/deletes only modify the corresponding partitions





Oversized partition

Leads to read amplification for some queries, tail latency increases

Static centroids

Does not represent dynamic data-sets well, accuracy drops

Strawman solution for update

- Periodic global index rebuild
  + Refresh performance & accuracy immediately after rebuild

Strawman solution for update

- Periodic global index rebuild
  + Refresh performance & accuracy immediately after rebuild
  - **Deteriorating** performance & accuracy between rebuilds
  - Global rebuild is prohibitively **expensive**

	Memory	**CPU**	**Time**
DiskANN[1]	1100 GB	32 Cores	2 days
	64 GB	16 Cores	5 days
SPANN[2]	260 GB	45 Cores	4 days

Billion-scale rebuild cost

[1] Subramanya et al. DiskANN: Fast Accurate Billion-Point Nearest Neighbor Search on a Single Node. (NeurIPS 2019)
[2] Chen et al. SPANN: Highly efficient Billion-scale Approximate Nearest Neighbor Search. (NeurIPS 2021)

Strawman solution for update

- Periodic global index rebuild
  - \+ Refresh performance & accuracy immediately after rebuild

Is it possible to update vector index without expensive global rebuild?

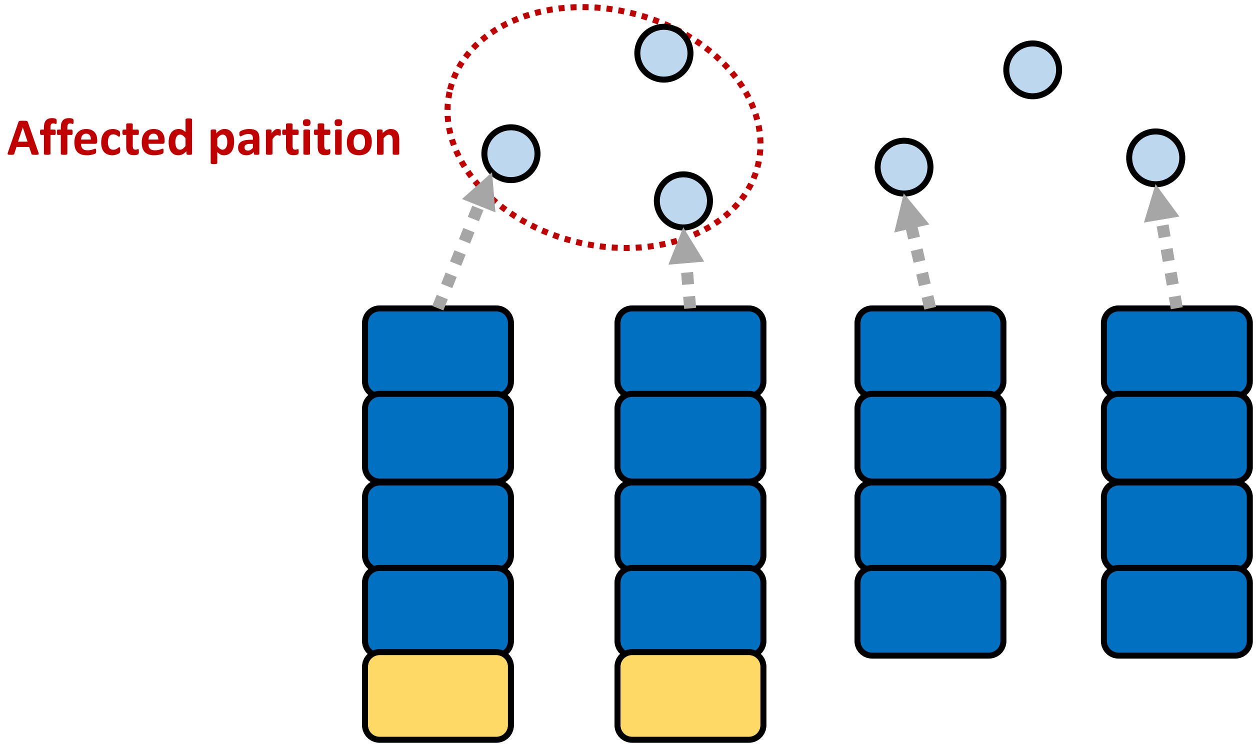
	64 GB	16 Cores	5 days
SPANN[2]	260 GB	45 Cores	4 days

Billion-scale rebuild cost

[1] Subramanya et al. DiskANN: Fast Accurate Billion-Point Nearest Neighbor Search on a Single Node. (NeurIPS 2019)
[2] Chen et al. SPANN: Highly efficient Billion-scale Approximate Nearest Neighbor Search. (NeurIPS 2021)

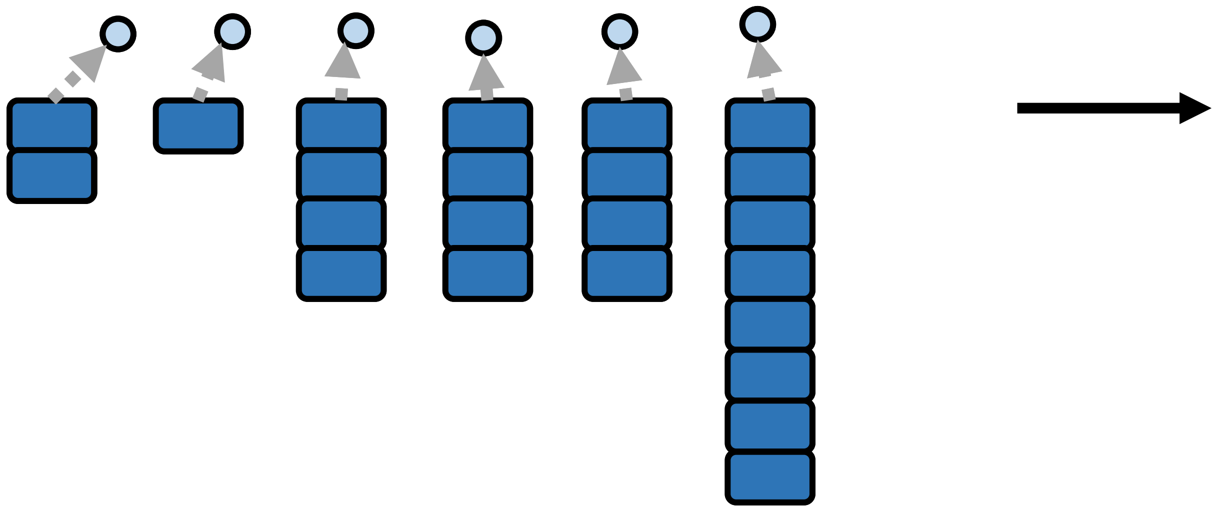
Recap: coarse-grained cluster-based index

- A silver lining towards fast vector updates
  - Small updates to a well-balanced index possibly incur local changes



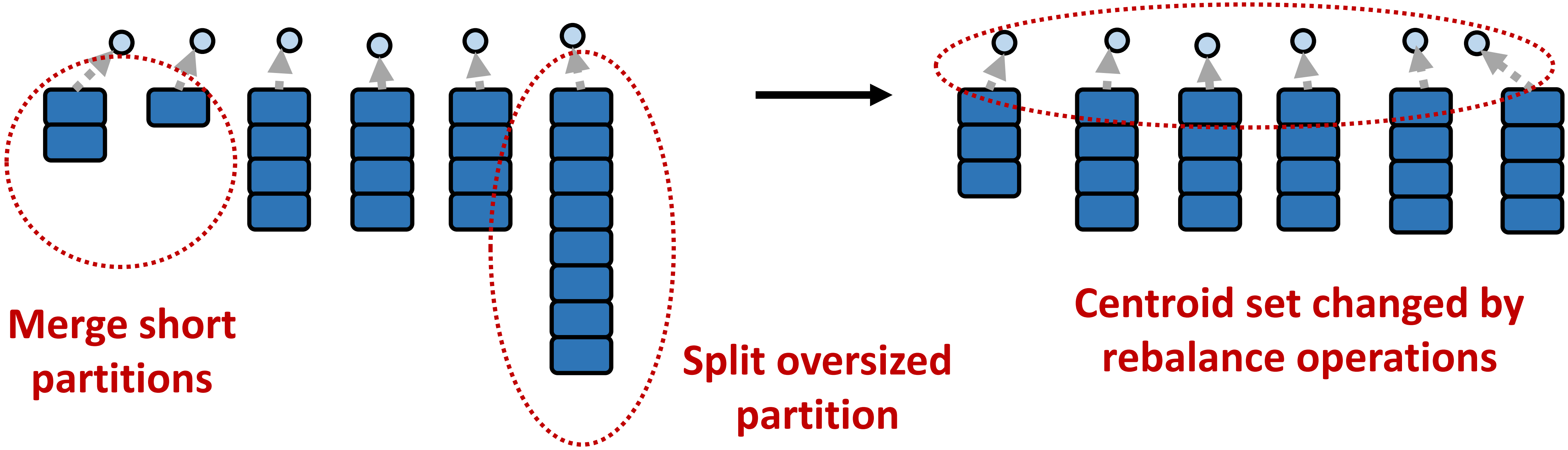
Idea: balance locally and incrementally

Split & merge



Split & merge

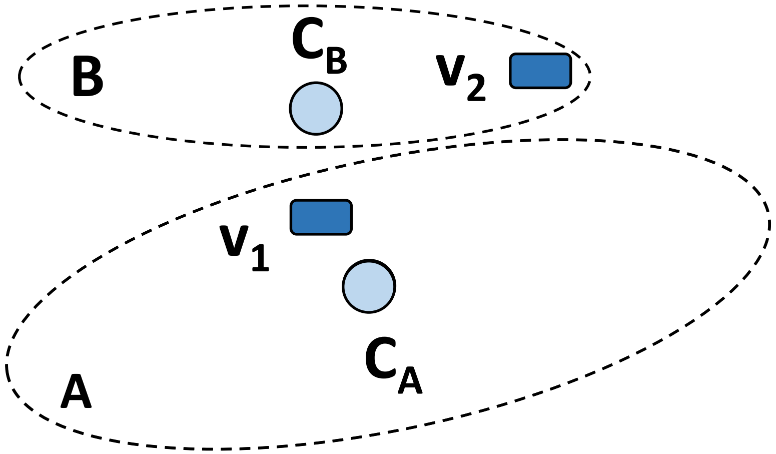
- Partition size imbalance → balance the partitions **locally**
- Static centroids → adapt shifting distribution **incrementally**



But such operations break the property of vector index

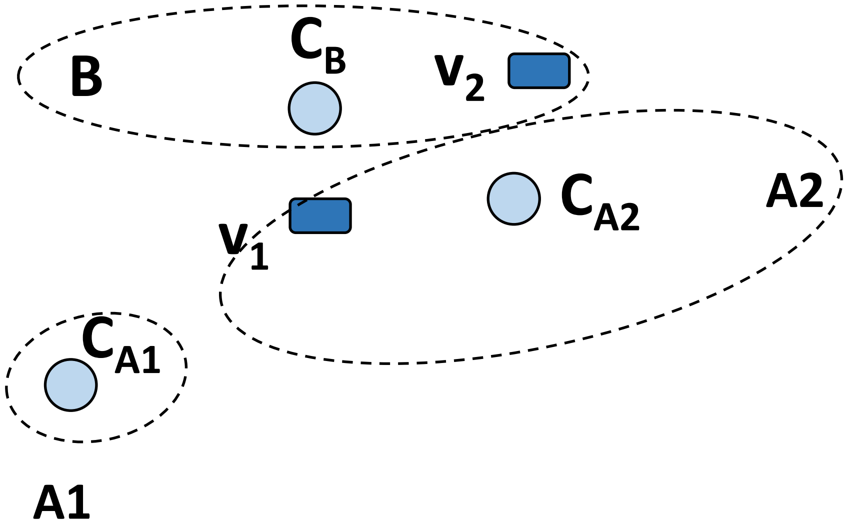
NPA: the key invariant for vector index

- **NPA (nearest partition assignment)**
  - Each vector should be put into the nearest partition



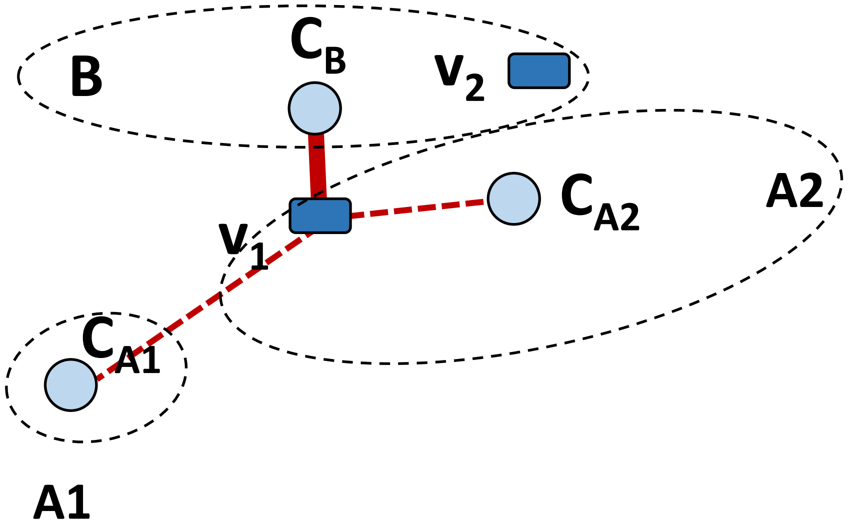
Split & merge violates NPA

- **NPA (nearest partition assignment)**
  - Each vector should be put into the nearest partition



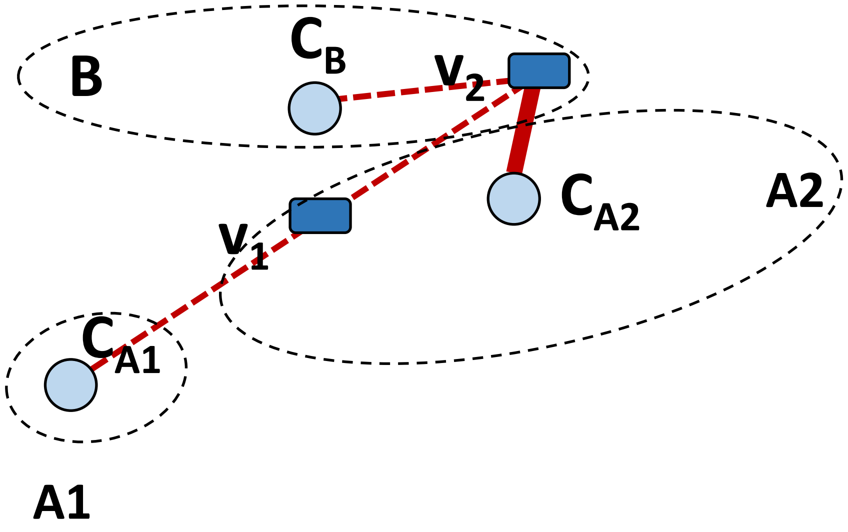
Split & merge violates NPA

- **NPA (nearest partition assignment)**
  - Each vector should be put into the nearest partition
- For v_1 in **split partition**, the closest centroid is **changed to** C_B



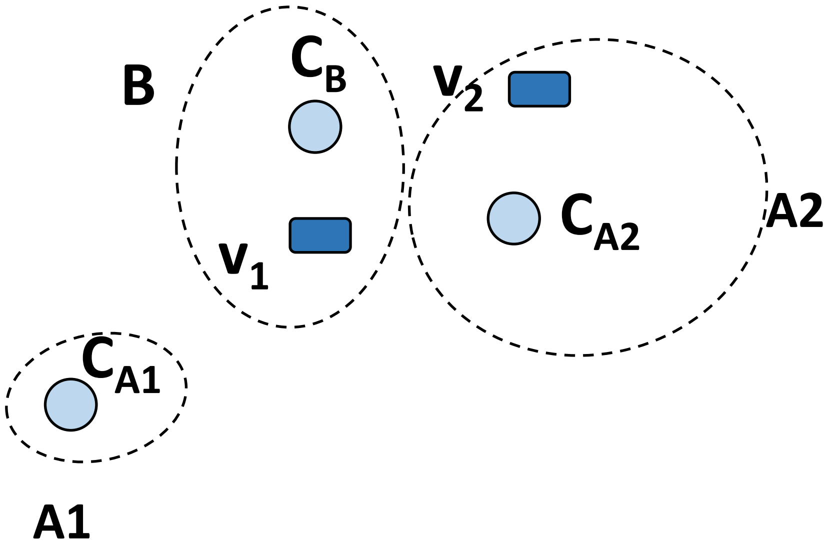
Split & merge violates NPA

- **NPA (nearest partition assignment)**
  - Each vector should be put into the nearest partition
- For v_2 in **unsplit partition**, the closest centroid is **changed to C_{A2}**



Reassigns are required to maintain NPA

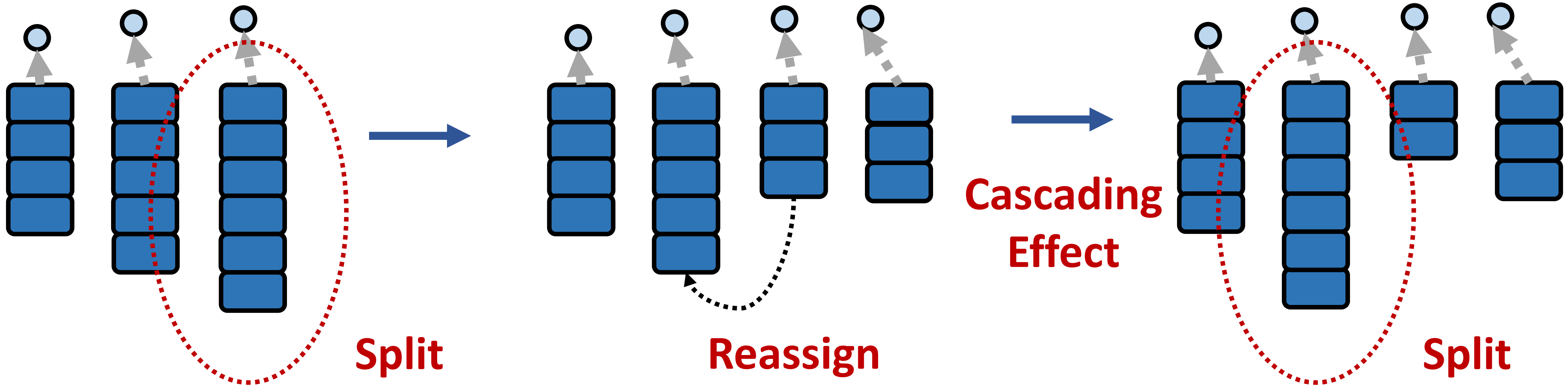
- But reassigning **all** vectors is expensive
  - Based on former cases, we derive **necessary conditions** to identify violations
  - Violation: closest partition != current partition



Section3.3 for detailed description of conditions

Cascading operations

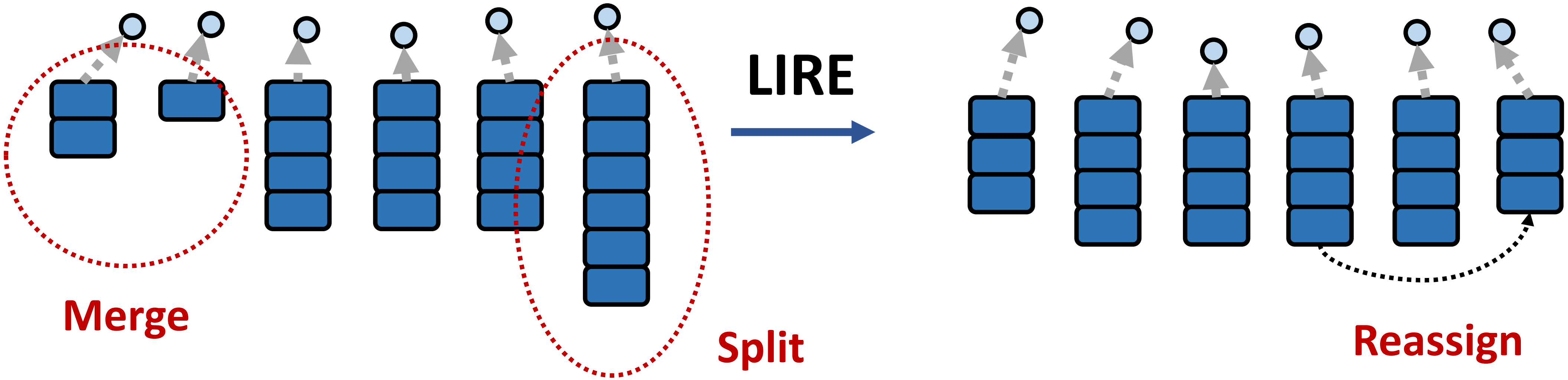
- Reassign may cause cascading split & merge
  - Incremental rebalance progress will converge
    - Reassignment ensures every vector receives closest centroid



Section3.4 for formal proof

LIRE Protocol

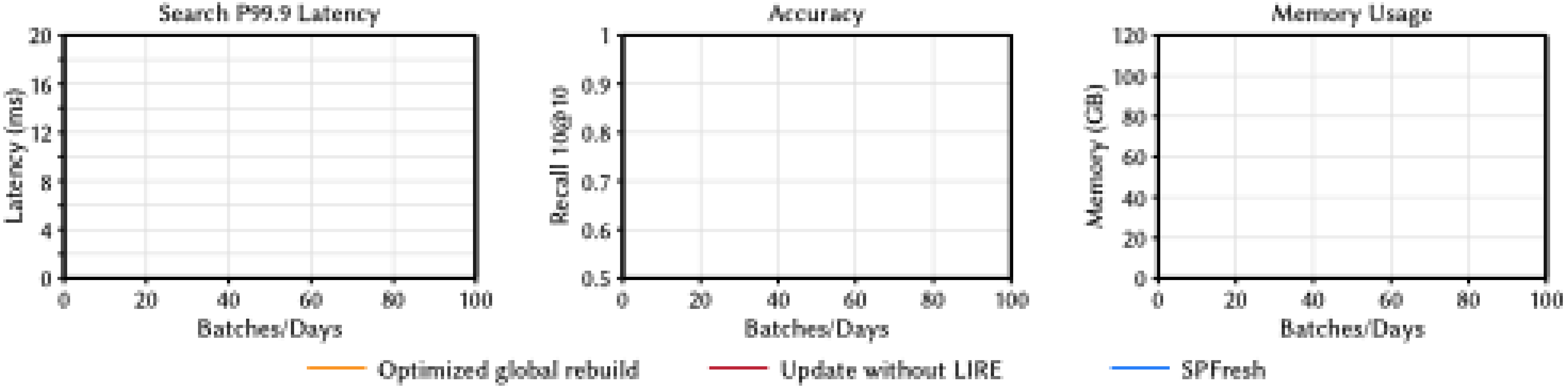
- LIRE: Lightweight Incremental RE-balancing
  - Keeps the size of every partition in balance → **Performance** ↑
  - Captures the change of data distribution → **Accuracy** ↑
  - Avoids global rebuild → **Resource** ↓



For more details, please refer to the paper!

Evaluation overview

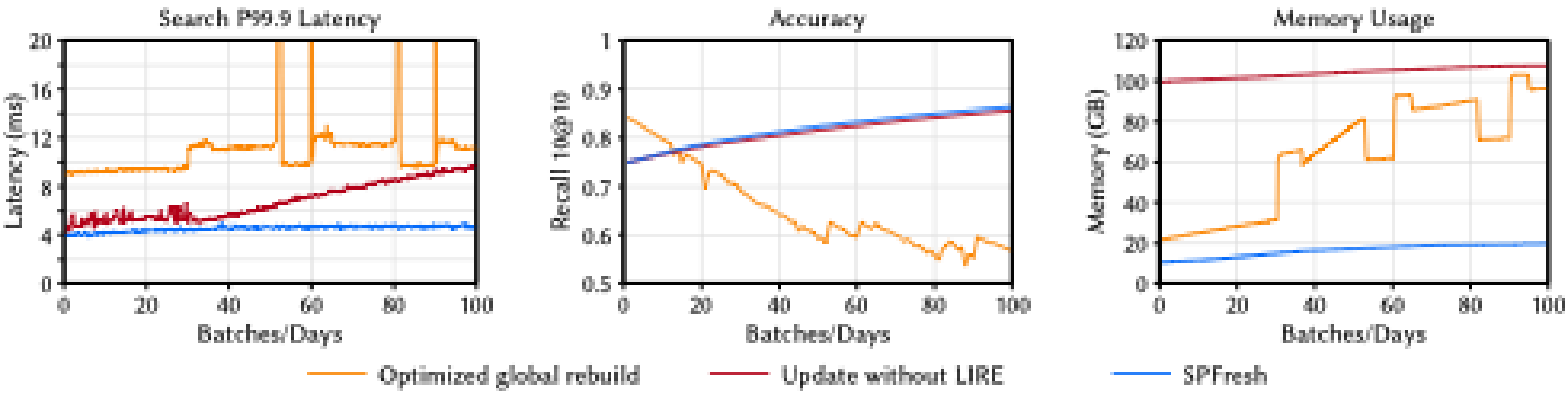
- Simulates a realistic vector update scenario with 1% daily update*



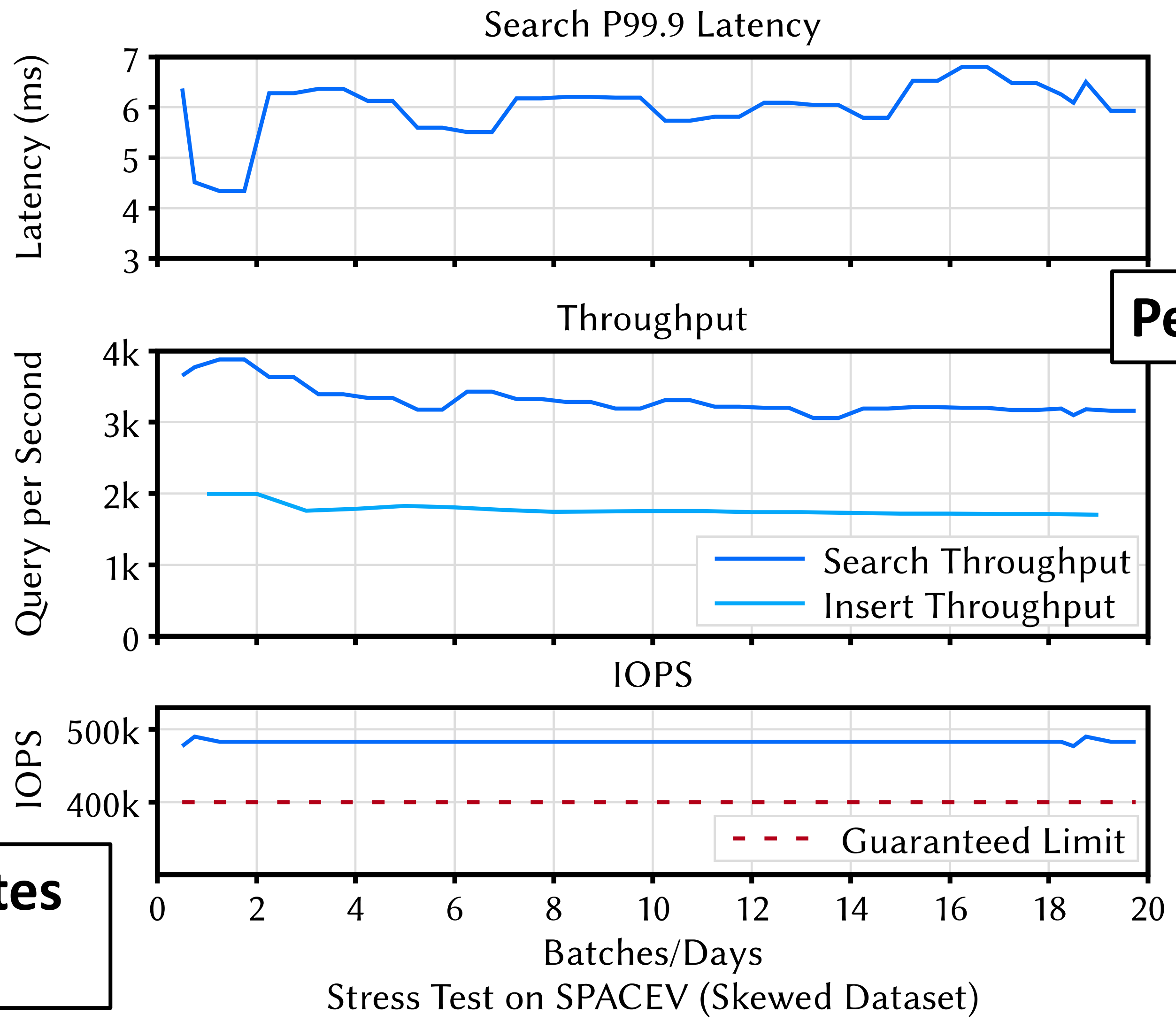
* Li et al. The Design and Implementation of a Real Time Visual Search System on JD E-Commerce Platform. (Middleware'18)

SPFresh performs well at update scenario

- Overall, SPFresh maintains **2.41X** lower P99.9 latency than baselines.
- SPFresh **keeps in high accuracy**, handles dynamic scenario
- SPFresh achieves as low as **5.30X** memory usage than baselines



SPFresh scales to billion-level scenario



Performs stably

With only extra 10GB DRAM for updating

Fully saturates SSD's IOPS

Takeaway from SPFresh

- We introduce SPFresh, a system that supports in-place update for billion-scale vector search.
- LIRE allows to **locally and incrementally** rebalance the data partitions.
- SPFresh can incorporate continuous updates with **low resources** while maintaining **high search accuracy**.
- SPFresh serves in billion-scale update scenario with just **single machine**.

Thanks!