SeLeP: Learning Based Semantic Prefetching for Exploratory Database Workloads

Farzaneh Zirak

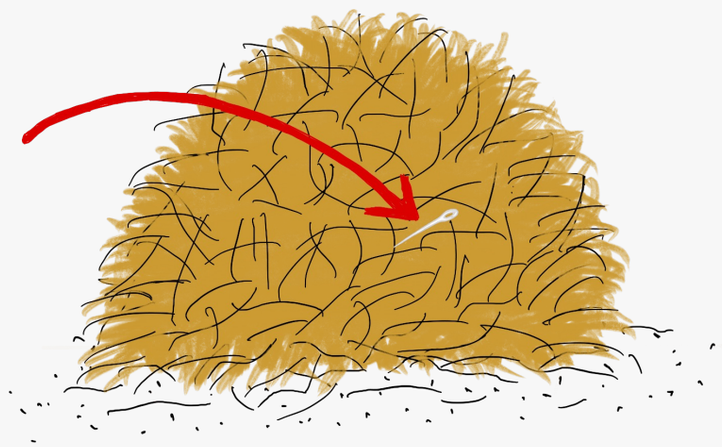
Supervisors:

Renata Borovica-Gajic, Farhana Choudhury

23-Apr-2024

data exploration

- Searching for insights buried in vast amount of data
- Data being collected at a striking rate



Rising need for effective data exploration

- Challenges and requirements:

☐ **No a priori knowledge**

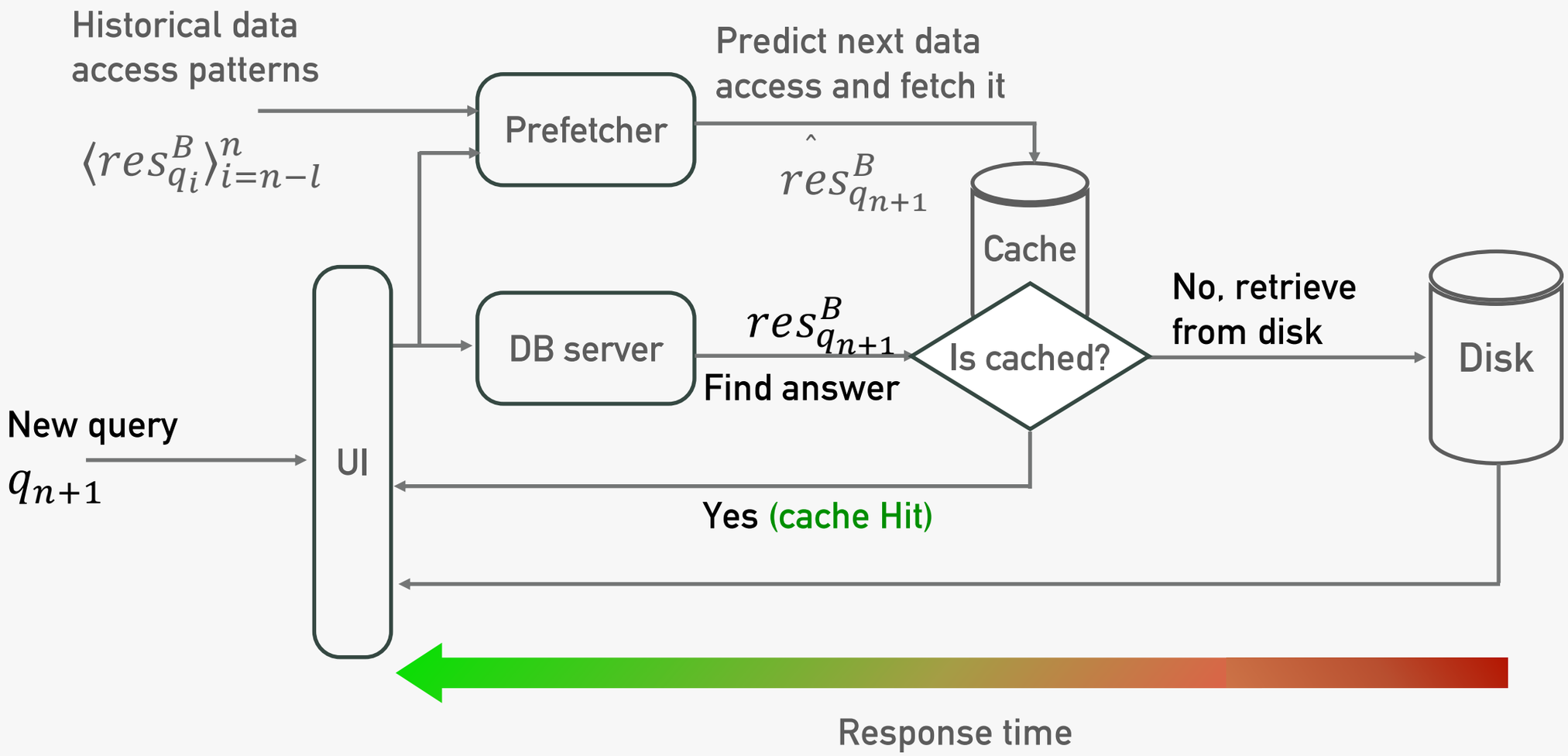
☐ **Interactive**

☐ **Non-expert user**

☐ **Adaptive**

Insight

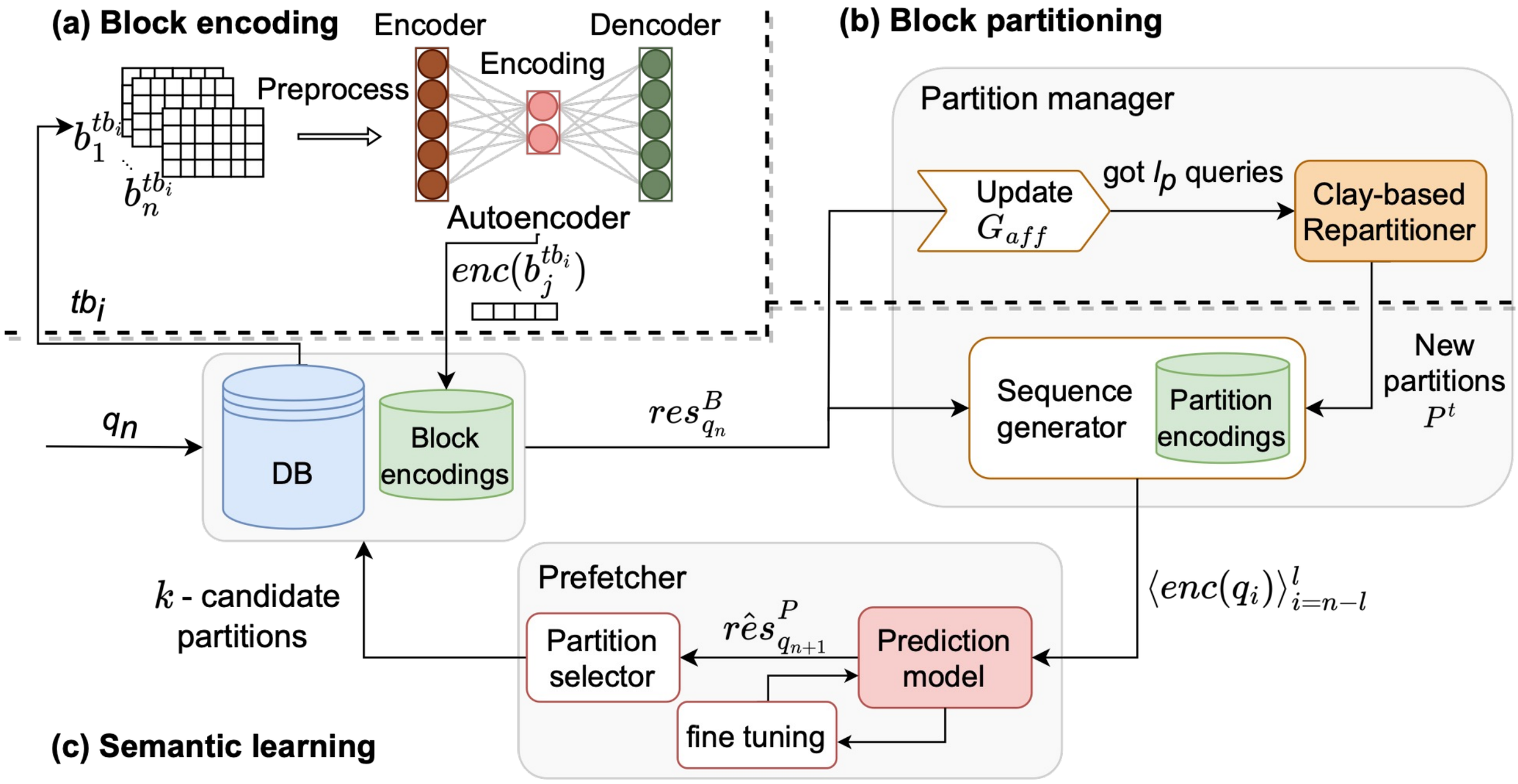
- Increase interactivity by reducing response time using **prefetching** technique
- Predict the upcoming workload and fill the cache efficiently with data to be accessed



Objectives

Semantic based	Make prefetch decision considering actual data values (instead of using Logical Block Addresses)
Suitable for various workloads	Support both navigational and SQL-based workloads
Adaptive	Can adapt to workload shifts

SeLeP Overview [VLDB'24]

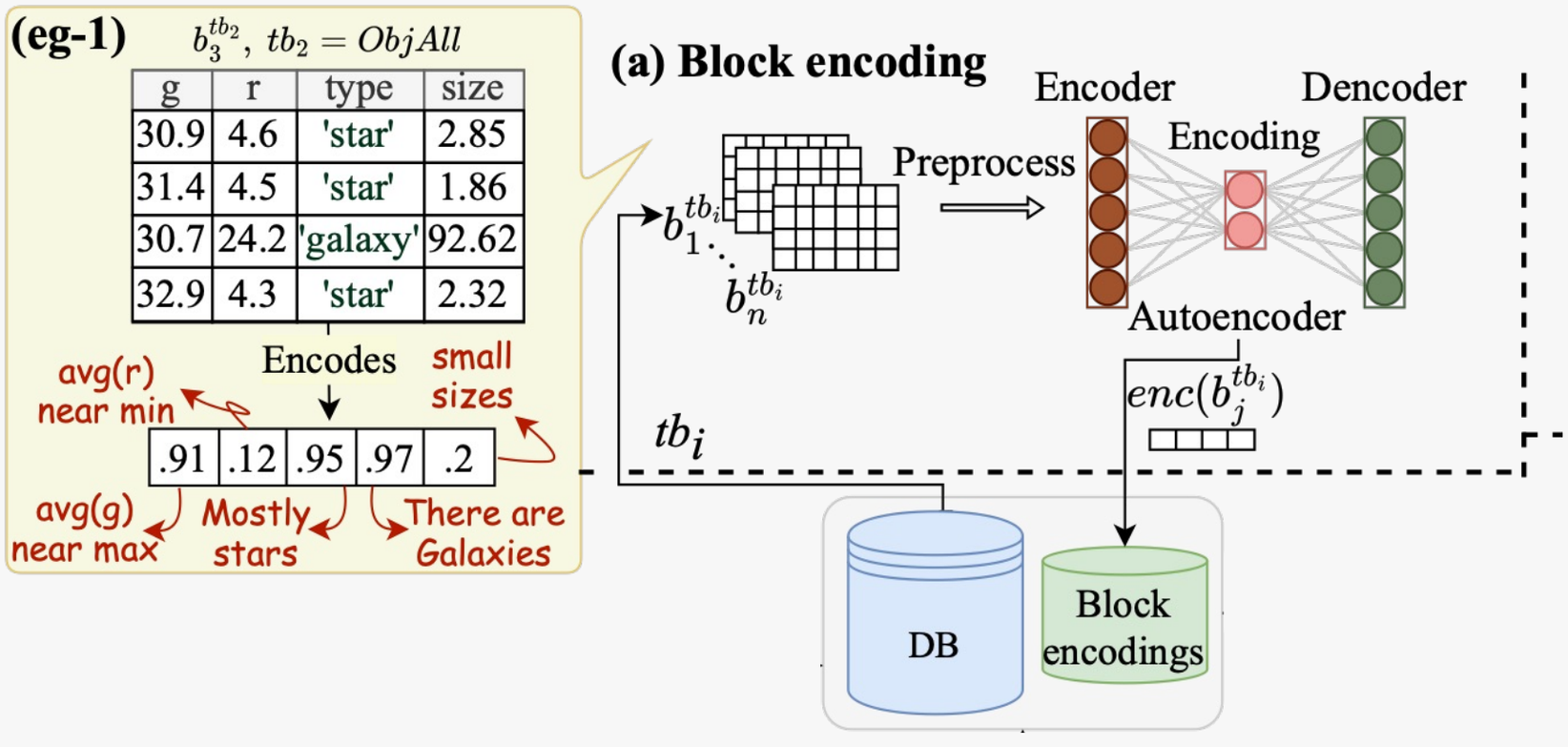


Block encoding

Semantic based

- Each block contains hundreds of values
- Cannot detect the connection between data values in a sequence of blocks
- Need a concise representation for each block which captures the essential and distinctive characteristics of its data

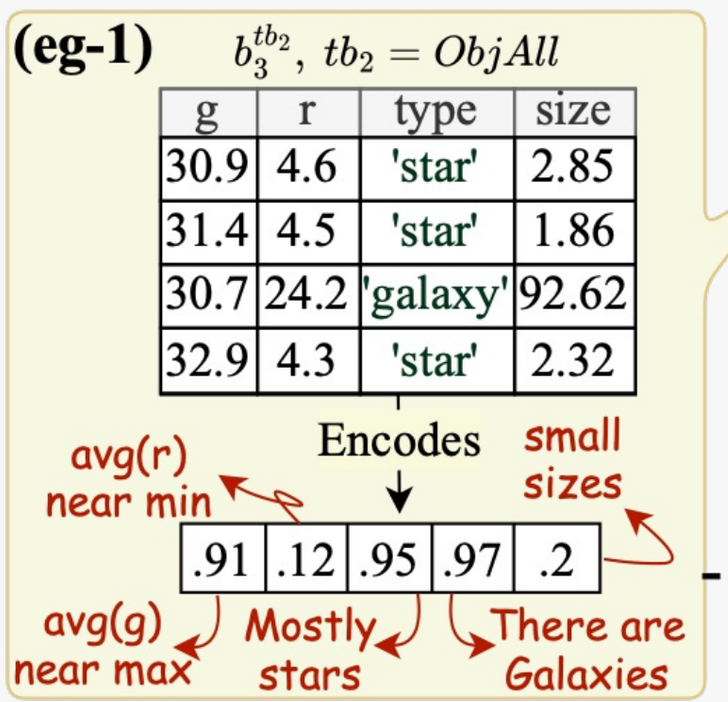
Consider each block of data as a big matrix and encode them into a vector of length l_b using Autoencoders

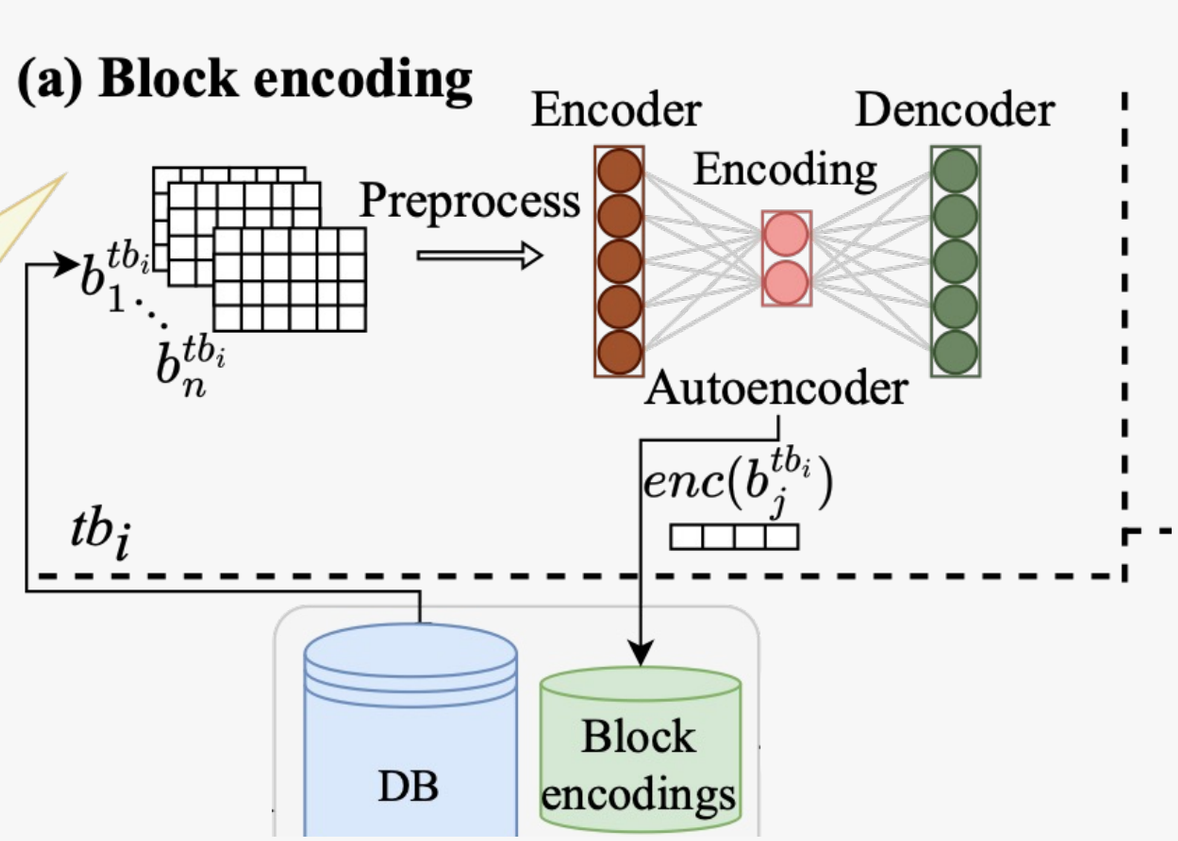


Block encoding

Semantic based

- Encoder and decoder have same layer structure consisting of two dense layers
- Preprocessing includes dimensionality reduction using PCA, string to number conversion using word2vec, and data normalisation





Query encoding

Support various workloads

- Visual queries are easy, a few adjacent blocks get accessed

- SQL queries are harder to predict, a set of blocks are getting accessed with various size from different part of the data space

- Need a better representation for queries:

  Represent queries by aggregated semantics of its accessed data

- Instead of directly using the blocks $\langle res_{q_i}^B \rangle_{i=n-l}^{n}$, we can input a sequence of encodings

- Using query encodings, the semantic prefetching problem can be transferred to a time series forecasting problem using a multi-label classification model:

  query encoding sequence $\langle enc_{q_i} \rangle_{i=n-l}^{n}$ as input and $res_{q_{n+1}}^B$ as the output

Query encoding

Support various workloads

- Visual queries are easy, a few adjacent blocks get accessed

- SQL queries are harder to predict, a set of blocks are getting accessed with various size from different part of the data space

- Need a better representation for queries:

  Represent queries by aggregated semantics of its accessed data

- Instead of directly using the blocks $\langle res_{q_i}^B \rangle_{i=n-l}^{n}$, we can input a sequence of encodings

- Using query encodings, the semantic prefetching problem can be transferred to a time series forecasting problem using a multi-label classification model:

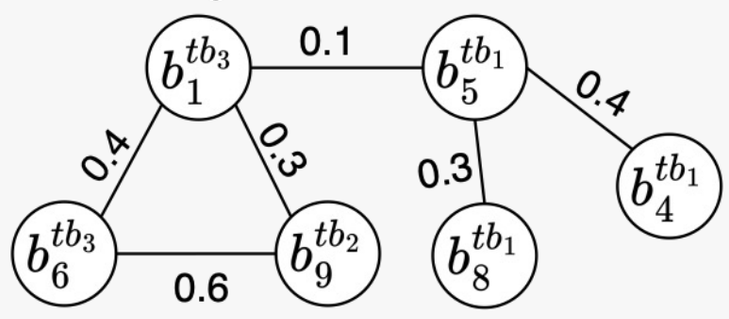
  query encoding sequence $\langle enc_{q_i} \rangle_{i=n-l}^{n}$ as input and $res_{q_{n+1}}^B$ as the output

Substantial number of labels in the classification model

Block partitioning

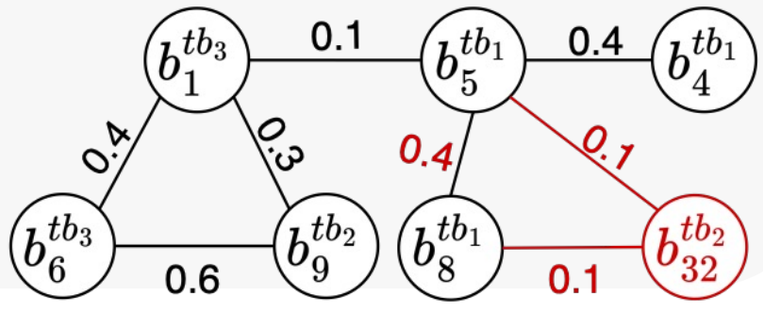
- To reduce number of classes, we cluster blocks frequently accessed together into partitions, which are more sizeable sets
- Construct affinity graph G_{aff}, with nodes representing the blocks and edge weights illustrating co-accessing rate
- Cluster nodes using graph partitioning and load balancing methods
- Employed Clay partitioning which tries to minimize the inter cluster edges

Previous G_{aff} :



$res_{q_n}^{B}$: $b_5^{tb_1}$ $b_{32}^{tb_2}$ $b_8^{tb_2}$

Updated G_{aff} with l_p = 10 :



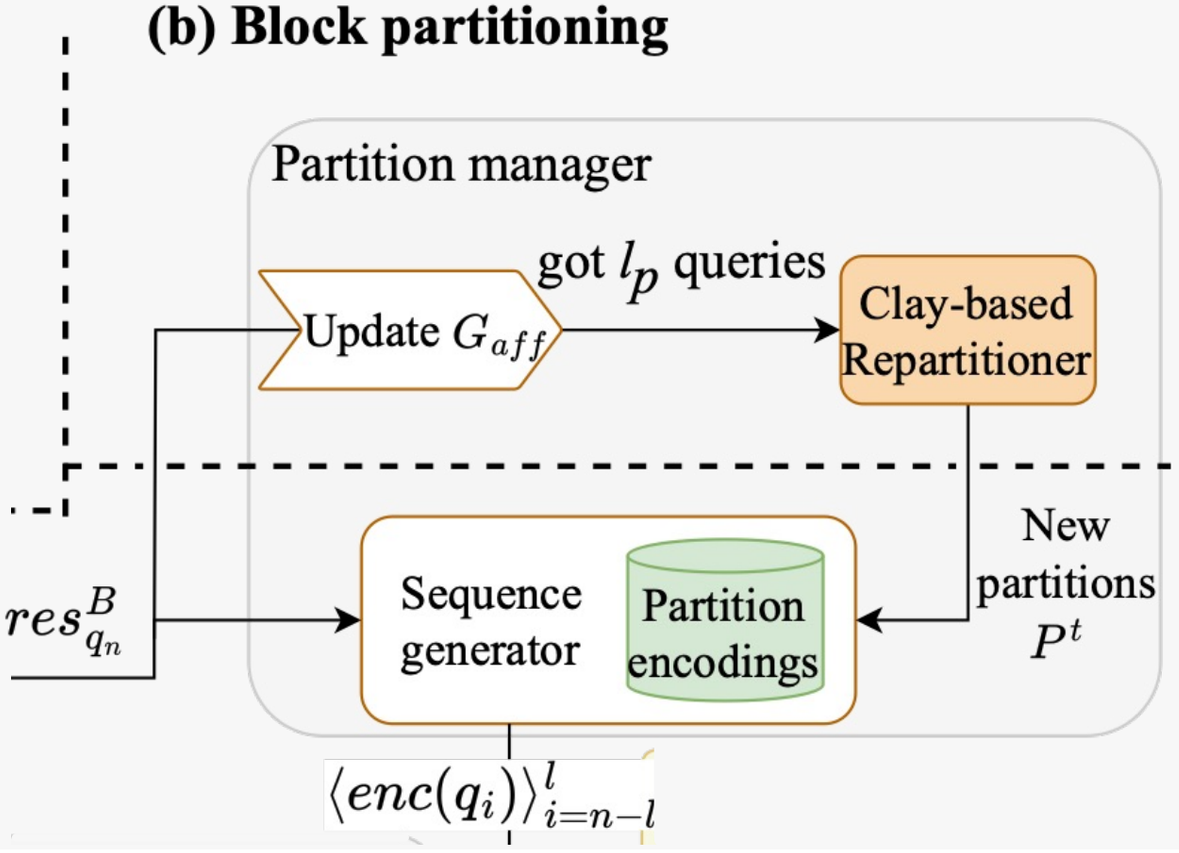
Block partitioning

- Aggregate block encodings to calculate partition encodings

tb_1	0	0	0	0	0
tb_2	.89	.21	.79	.68	.24
tb_3	0	0	0	0	0
tb_4	.4	.13	.33	.19	.81

- Instead of block address, use the partition number to refer to a data access

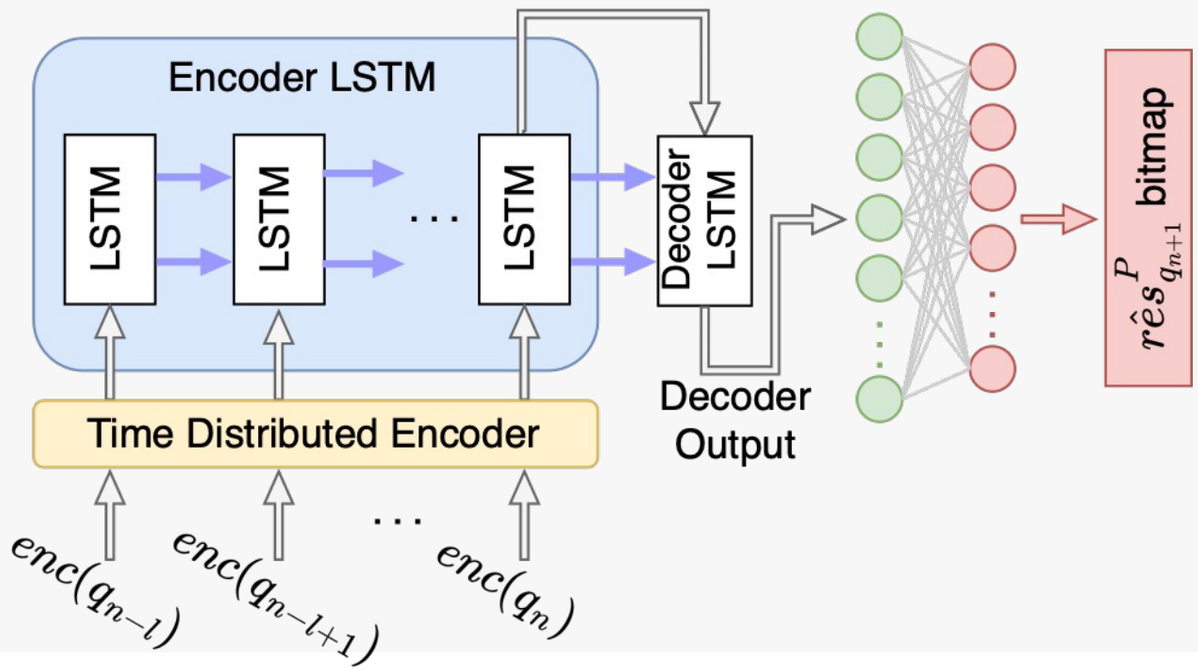
- Now we can input sequence of query encoding $\langle enc_{q_i} \rangle_{i=n-l}^{n}$ calculated by aggregating the partition encodings and get $res_{q_{n+1}}^{P}$ as the output



Semantic learning

Adaptive

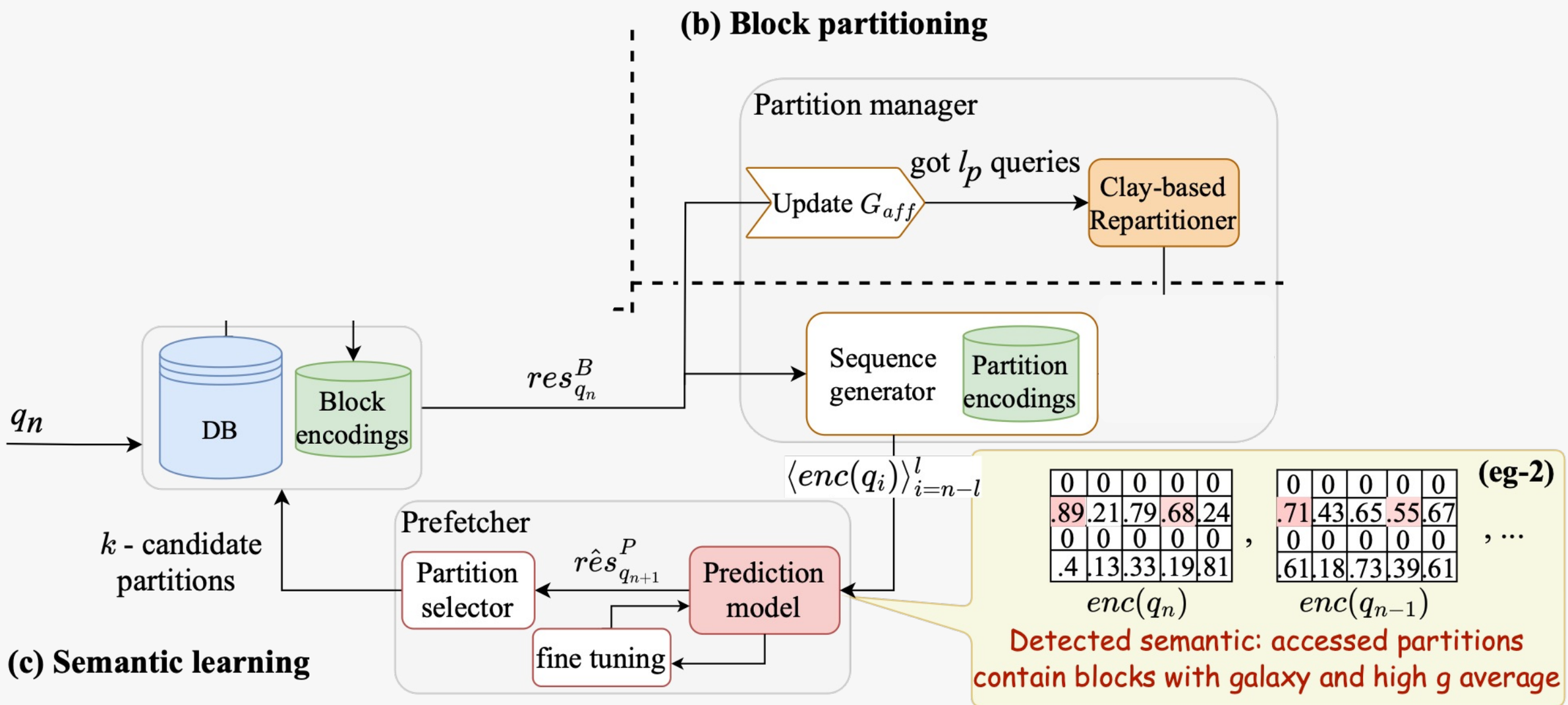
- Learn data access pattern and fine tune the model with new workloads
- The sequence of query encoding matrices are compressed via a time-distributed layer with 128 units
- The compressed vectors are fed into the encoder and decoder, both designed with a single-layer LSTM with 32 units
- The decoder's output further passes through two dense layers, each containing $|P|$ units
- The final dense layer generates a bitmap vector with a length of $|P|$, representing the probabilities of each partition being accessed in the subsequent query.



Semantic learning

Adaptive

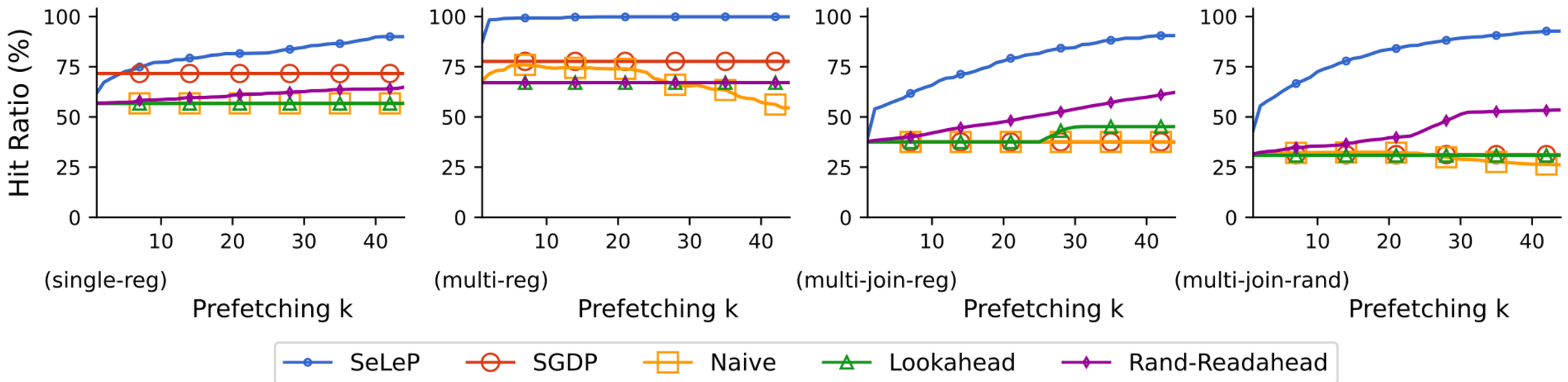
Continues repartitioning and fine tuning using the most recently received workload.



SeLeP in Action – Hit Ratio

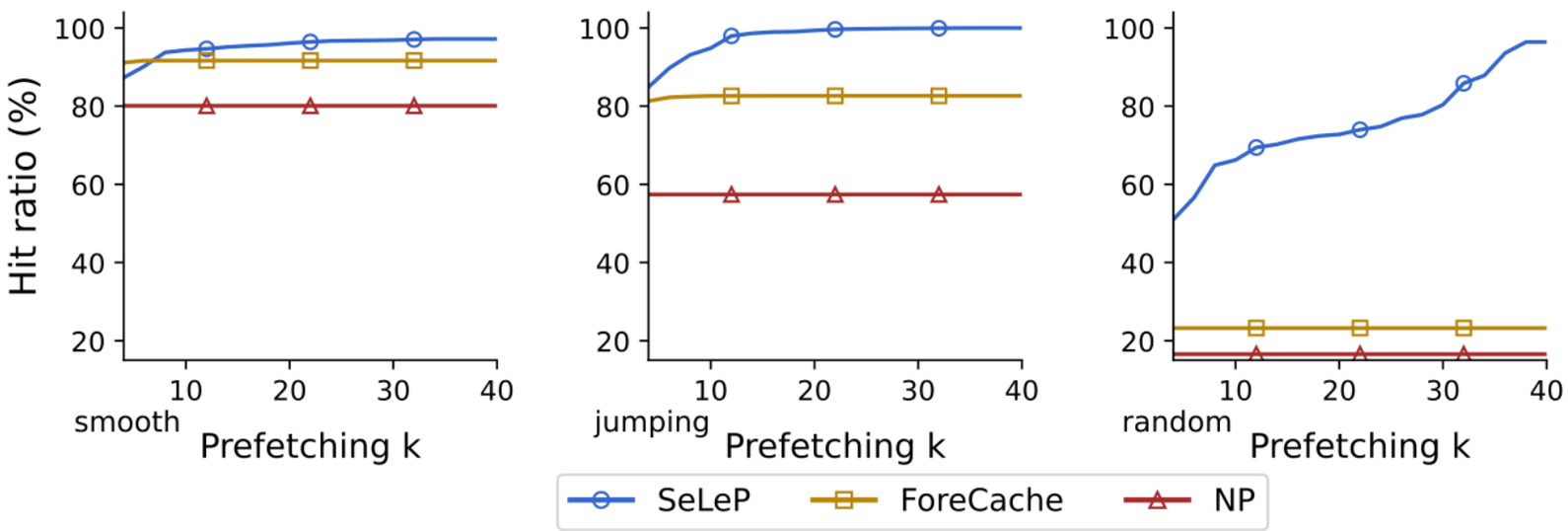
Setting: 16GB SDSS DR7, prefetch size = $k \times 128$ block, 4GB cache

Queries: Predictable(reg) and unpredictable(rand) single-table, multi-table, and containing join SQL workloads



Setting: 8GB SDSS DR7, prefetch size = $k \times 64$ block, 2GB cache

Queries: Smooth walk, jumping and random navigational workloads



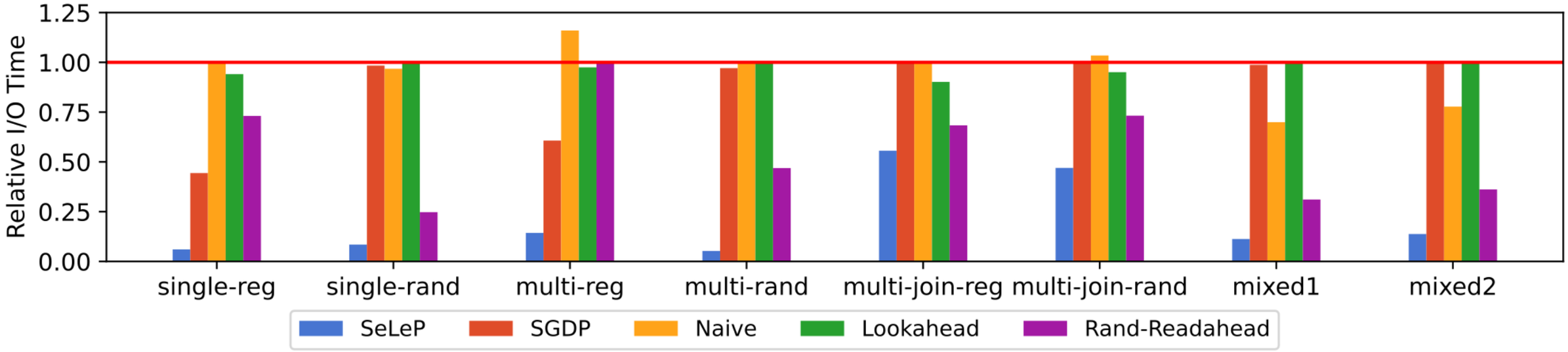
$$\text{Hit ratio} = \frac{\#hits}{\#hits + \#misses}$$

- **95%** hit ratio on **average**
- Up to **40% improvement** compared to the traditional and state-of-the-art prefetchers

SeLeP in Action – I/O Time

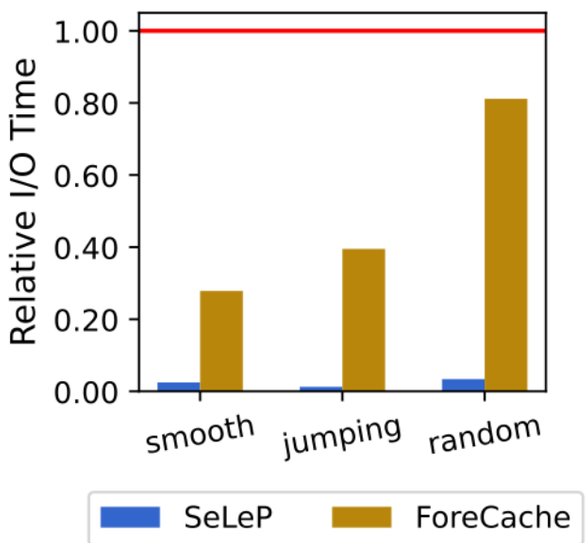
Setting: 16GB SDSS DR7, prefetch size = $k \times 128$ block, 4GB cache

Queries: Predictable(reg) and unpredictable(rand) single-table, multi-table, and containing join SQL workloads



Setting: 8GB SDSS DR7, prefetch size = $k \times 64$ block, 2GB cache

Queries: Smooth walk, jumping and random navigational workloads

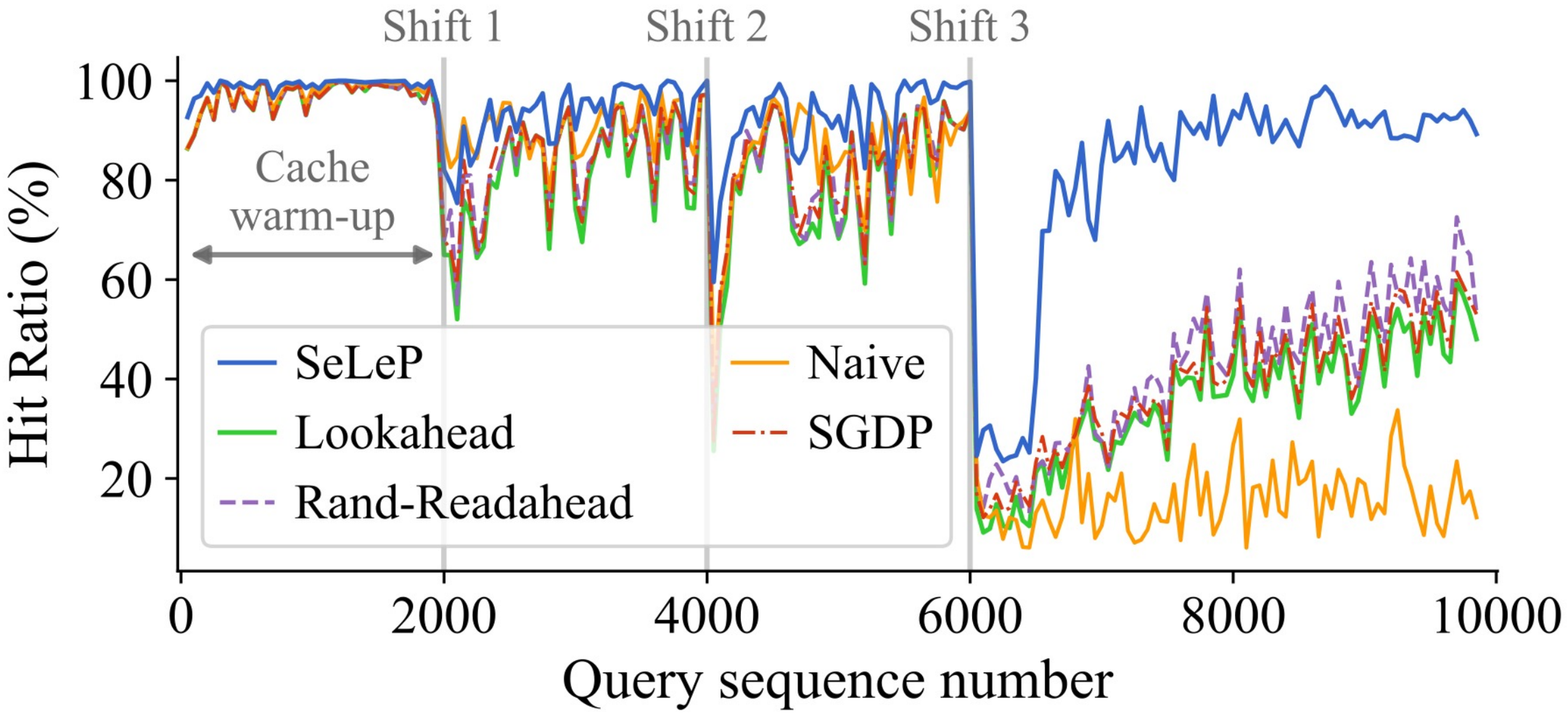


- **80%** I/O time reduction on **average**
- Up to **45% improvement** compared to the traditional and state-of-the-art prefetchers

SeLeP in Action – Adaptivity

Setting: 16GB SDSS DR7, prefetch size = $k \times 128$ block, 4GB cache

Queries: Shifts at sequence num = {2000, 4000, 6000} with novel query template and new accessed data



Summary of SeLeP

- Encodes blocks into vectors and **extract their semantics** using AutoEncoders
- **Dynamically partitions blocks** frequently accessed together
- **Learns partition access pattern** using LSTM model

Improves hit ratio up to 40% and **reduces I/O time** up to 45% compared to traditional and state-of-the-art prefetchers

Enables **high performance prefetching** for **SQL workloads**

Publication: *The paper proposing SeLeP is accepted to be published in VLDB 2024 conference.*

THE UNIVERSITY OF MELBOURNE

Questions?

Thank You