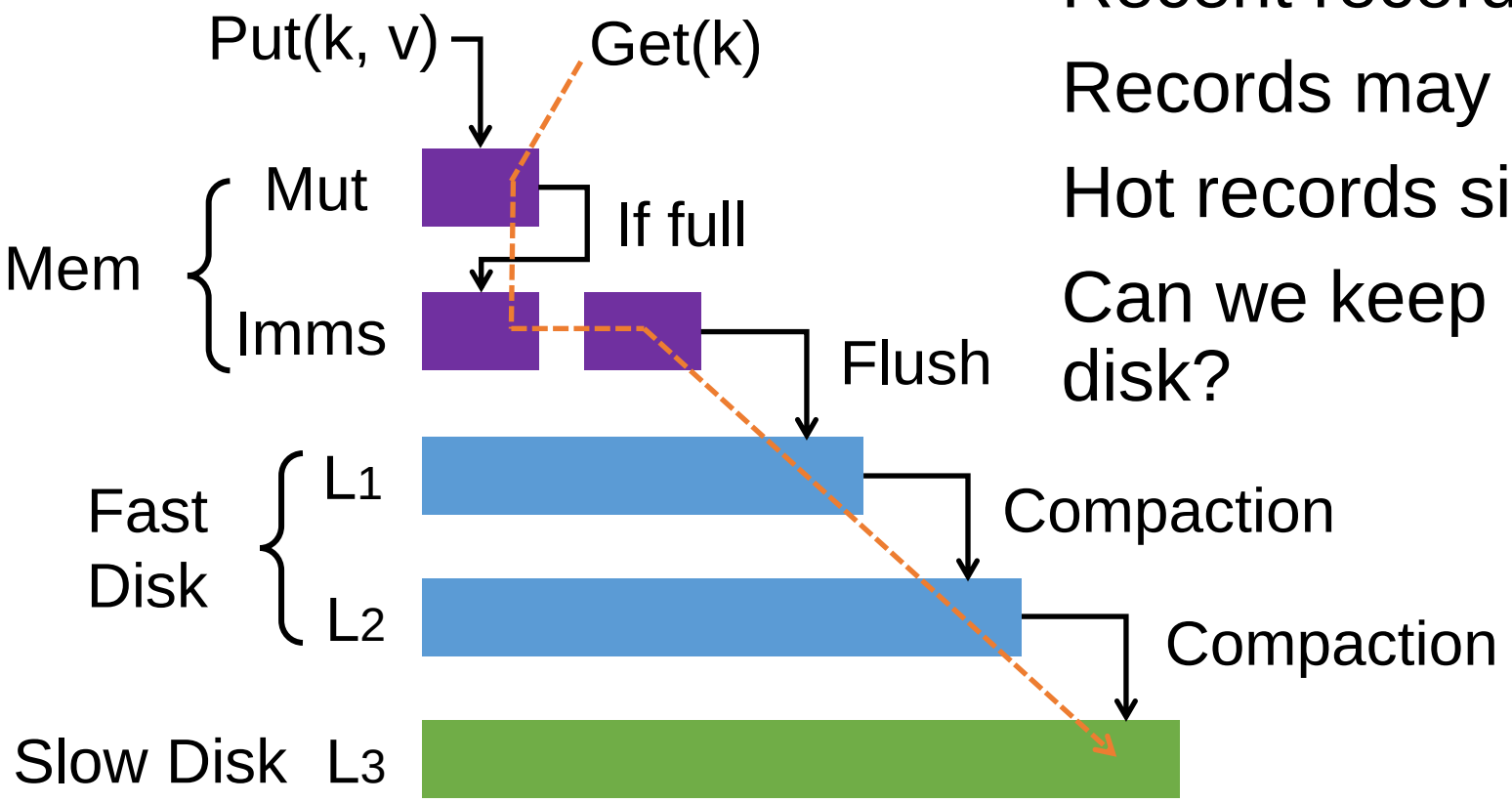
清華大學
Tsinghua University

HotRAP: Hot Record Retention and Promotion for LSM-trees with Tiered Storage

Jiansheng Qiu, Fangzhou Yuan, Mingyu Gao, Huanchen Zhang

Presenter: Yanqi Pan

LSM-trees with tiered storage



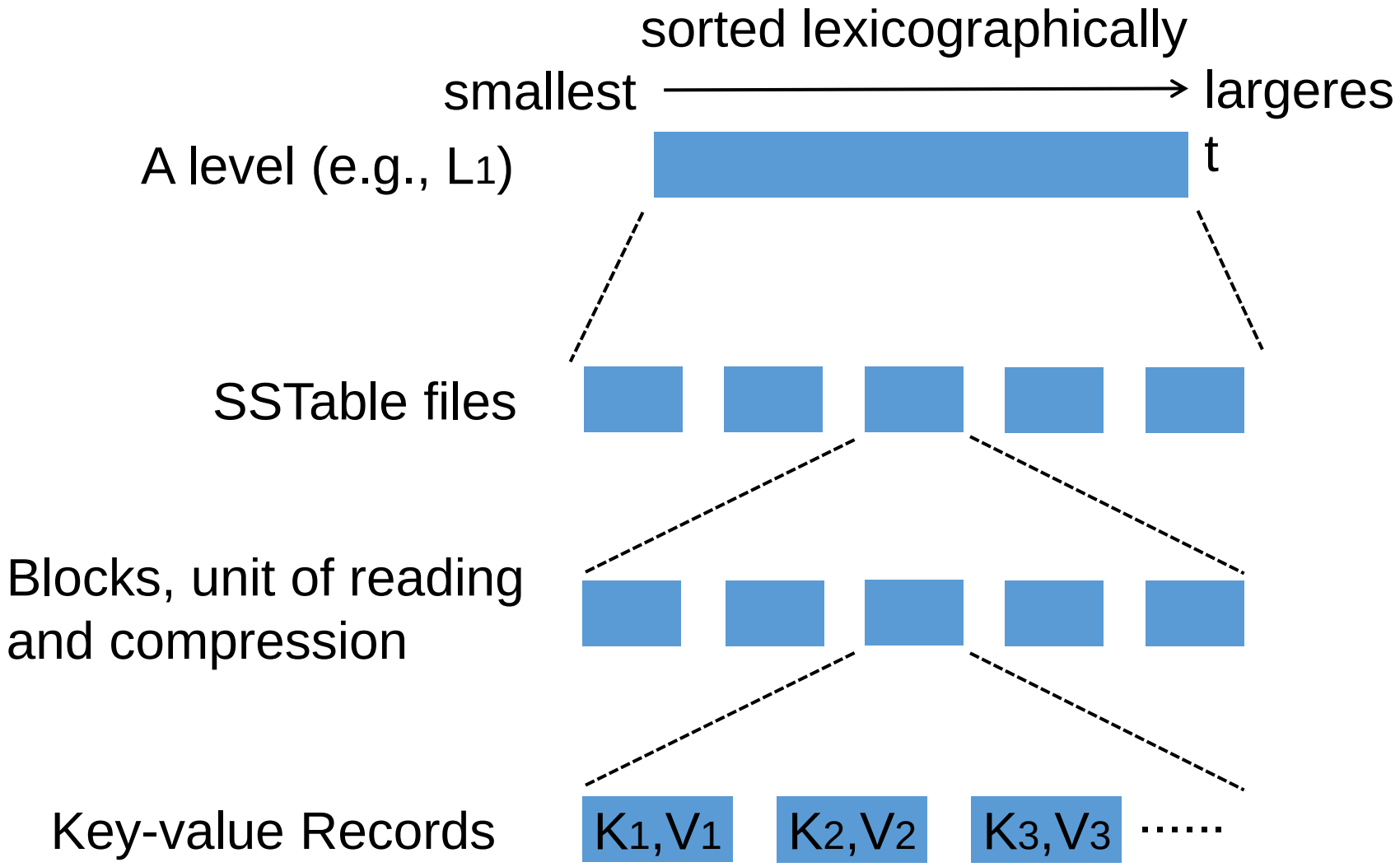
Recent records in upper levels (FD).

Records may stay hot for a long time.

Hot records sink into the slow disk.

Can we keep hot records in the fast disk?

Background: SSTables and blocks



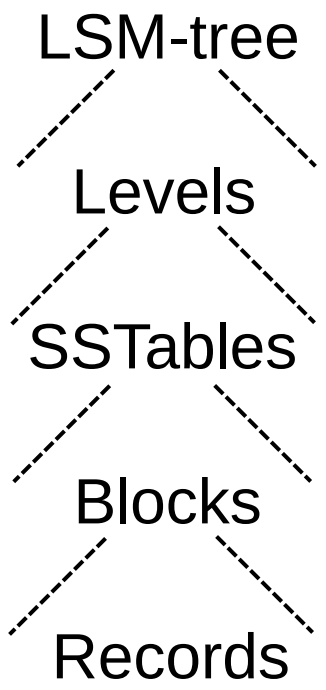
Previous works

Mutant: Stores hot SSTables in the fast disk.

SAS-Cache: Caches hot blocks in the fast disk.

MirrorKV:
- Splits the LSM-tree into the key and value LSM-tree.
- Caches hot key SSTables in the fast disk.
- Retains hot blocks in the fast disk during compactions.

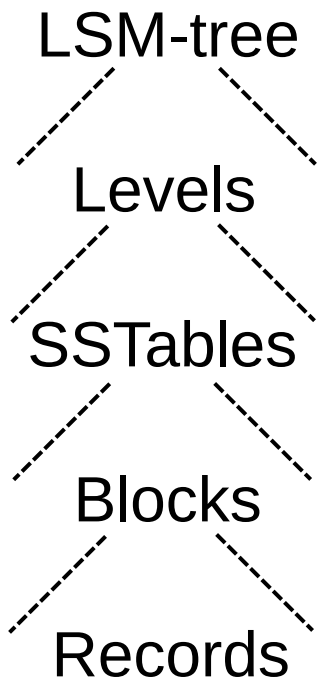
The granularity of SSTables and blocks are too coarse: cold records in a hot SSTable/block.



Previous works (record-level)

SA-LSM:
- Accurately predicts cold data with survival analysis.
- Demotes cold records to the slow disk.

Drawbacks: model too heavy, no promotion.

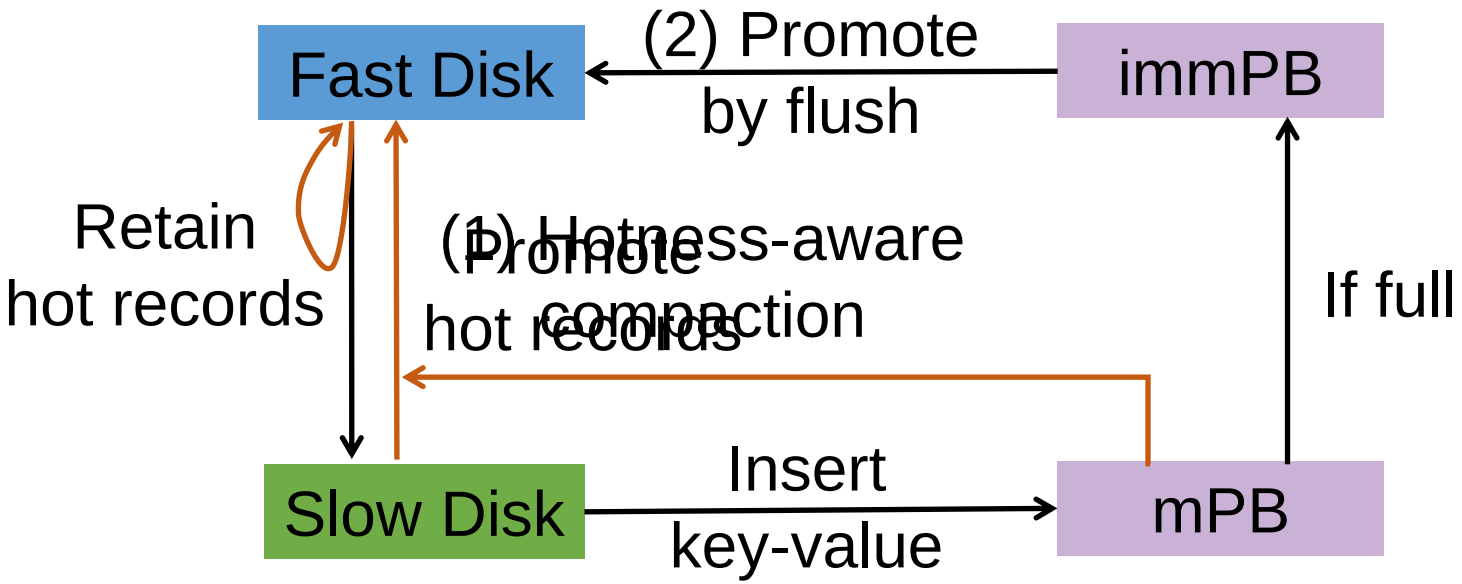


PrismDB:
- Estimates key popularity with the clock algorithm.
- Hot records are retained in or promoted to the fast disk during compactions.

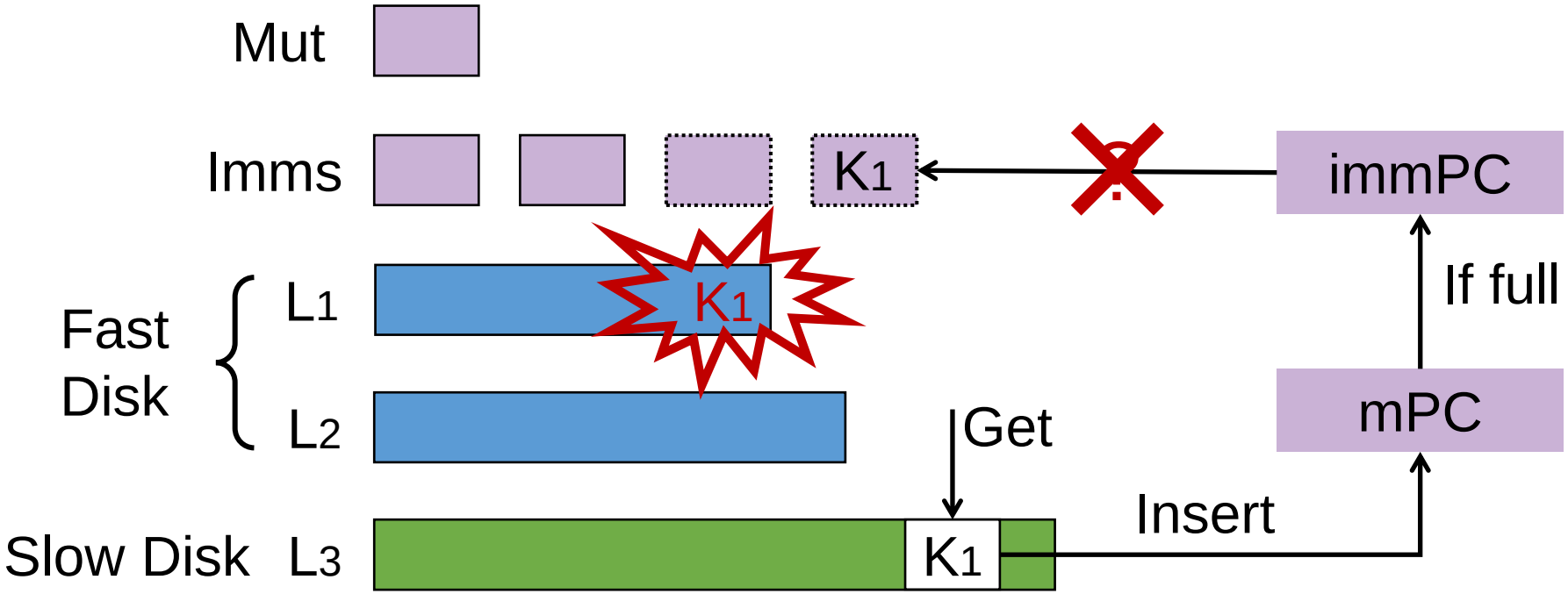
Drawbacks:
- Clock bits are indexed by a hash table: memory consuming.
- Promotes only during compactions. Inefficient if read-heavy.

HotRAP (Hot record Retention And Promotion)

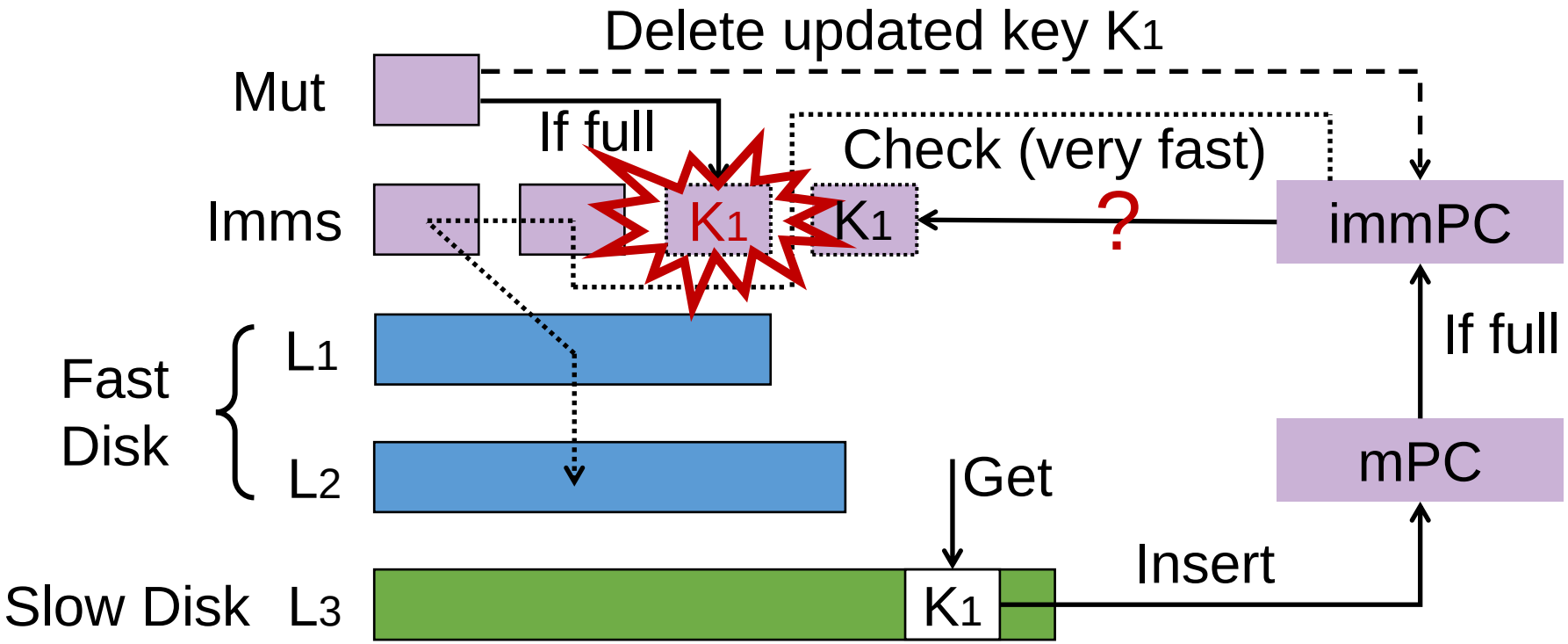
- Record-level
- Tracks hotness of records with on-disk RALT: save memory.
- Two pathways for hot records to reside in the fast disk:



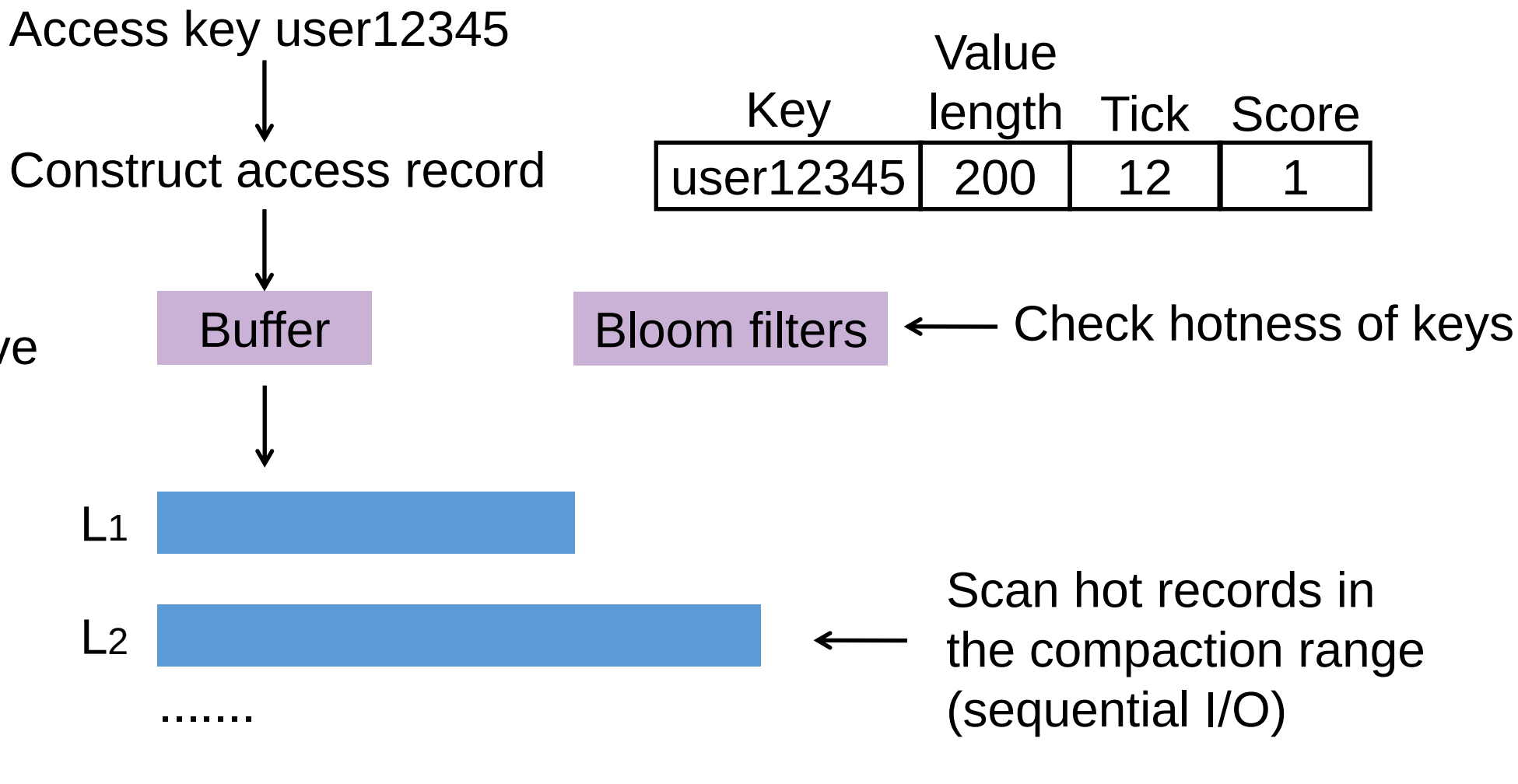
Promotion by flush (simplified)



Promotion by flush (simplified)

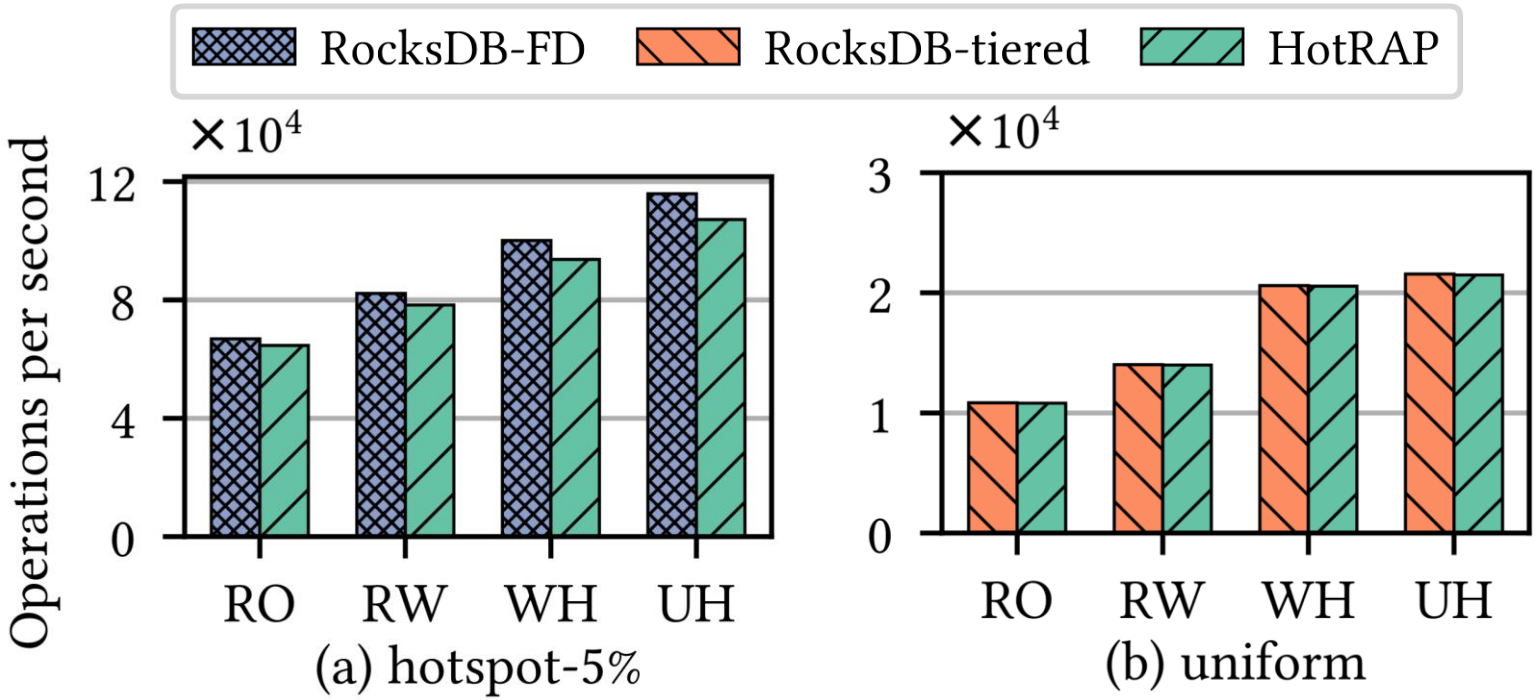


RALT: the hotness tracker of HotRAP



Evaluation

Notation	Meaning	Read-write ratio
RO	read-only	100% read
RW	read-write	75% read, 25% insert
WH	write-heavy	50% read, 50% insert
UH	update-heavy	50% read, 50% update



Conclusion

清華大學
Tsinghua University

- Record-level retention and promotion: doesn’t waste FD.
- On-disk hotness tracker: RALT, save memory.
- Promote by flush under read-heavy workloads.
- Negligible overhead under skew and non-skew workloads.

Read our paper for more details:

- The algorithm to automatically tune the size limit of RALT.
- Cost analysis of RALT: disk, memory, I/O
- Comparisons with other systems.
- Performance on real-world Twitter traces.