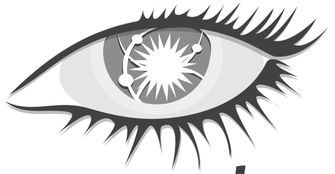
RangeReduce: Query-Driven LSM Compactions

Shubham Kaushik Manos Athanassoulis Subhadeep Sarkar

CockroachDB

RocksDB



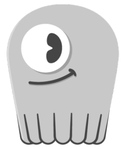
BigTable

OpenTSDB

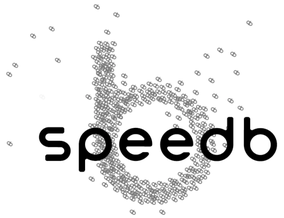
Key-Value Stores

Azure
Cosmos DB

DynamoDB



ScyllaDB



Brandeis
UNIVERSITY

BOSTON
UNIVERSITY

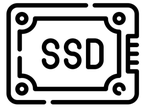

FoundationDB
RocksDB
SQLite
levelDB
CockroachDB
cassandra
APACHE HBASE
influxdb
Key-Value Stores
BigTable
OpenTSDB
speedb
Azure Cosmos DB
DynamoDB
TiKV
mongoDB
ScyllaDB
Brandeis UNIVERSITY
BOSTON UNIVERSITY
DiSC lab

Log-Structured Merge Trees

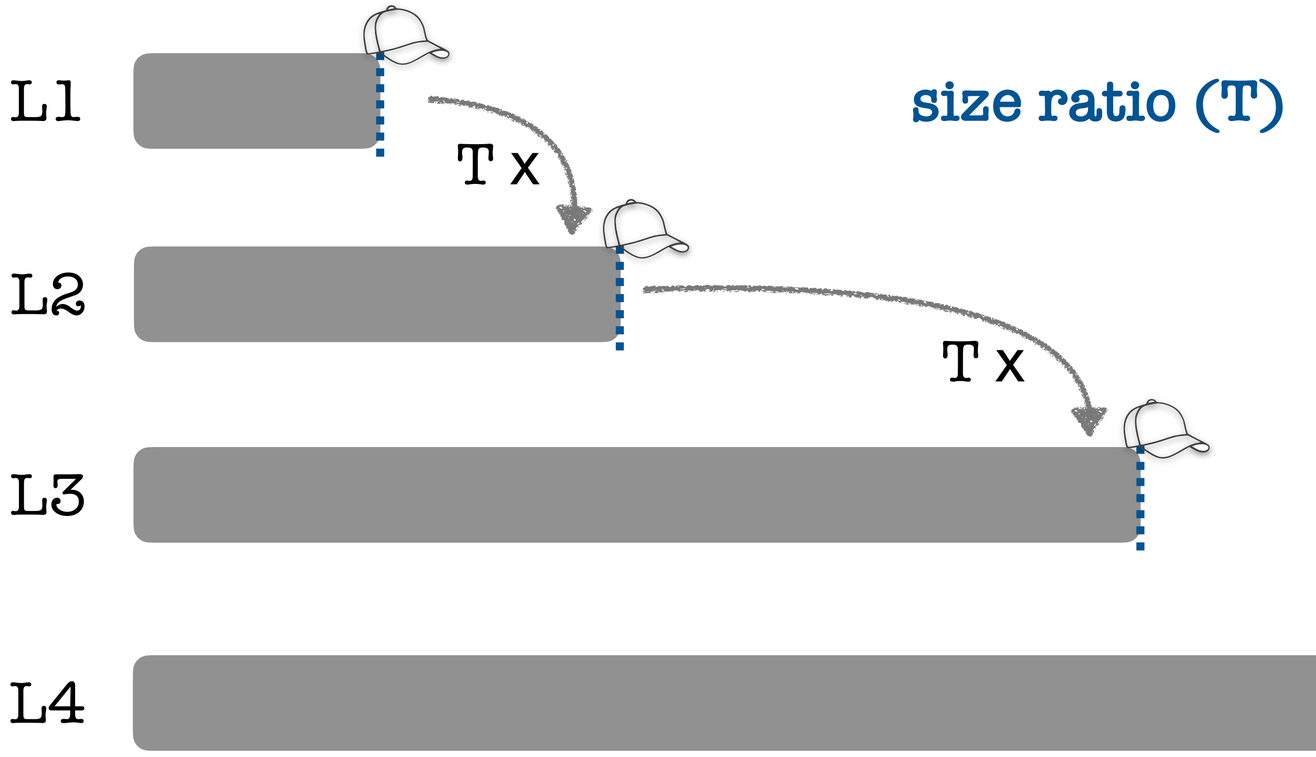
Brandeis
UNIVERSITY

BOSTON
UNIVERSITY

LSM **Overview**

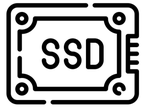


buffer

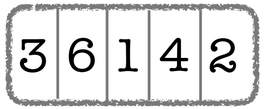


capacity grows exponentially

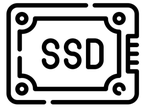
LSM Ingestion



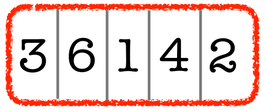
buffer



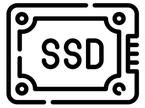
LSM Ingestion



buffer

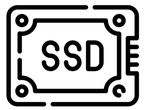


LSM Ingestion



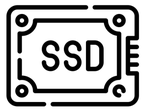
buffer

LSM Ingestion



buffer

LSM Ingestion

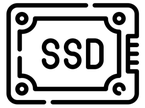


buffer

L1

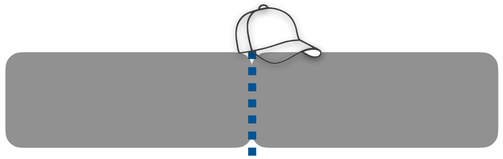
flush

LSM Ingestion

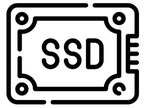


buffer

L1



LSM Compaction

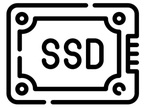


buffer

L1

compaction

LSM Compaction



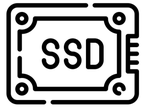
buffer

L1

L2

compaction

LSM Compaction



buffer

L1

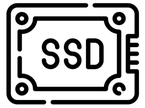
L2

L3

L4

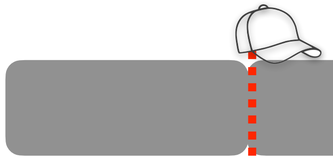
compaction

LSM Compaction



buffer

L1



L2

L3

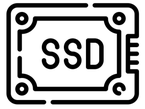
L4

Brandeis UNIVERSITY

BOSTON UNIVERSITY

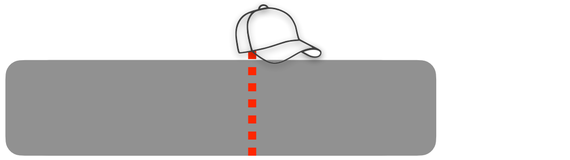
DiSC lab

LSM Compaction



buffer

L1

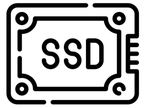


L2

L3

L4

LSM Compaction



buffer

L1

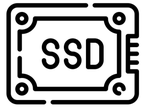
L2

L3

L4

cascading compaction

LSM Compaction



buffer

L1

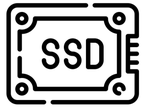
L2

L3

L4

cascading compaction

LSM Compaction



buffer

L1

L2

L3

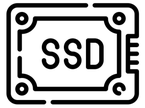
L4

cascading compaction

write stalls

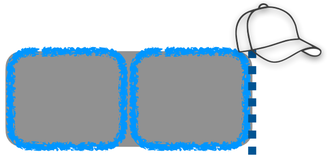
latency spikes

Partial Compaction in LSM



buffer

L1



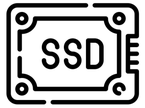
L2

L3

Brandeis UNIVERSITY

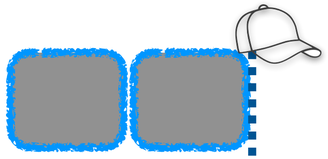
BOSTON UNIVERSITY

Partial Compaction in LSM



buffer

L1



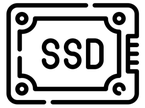
L2

L3

Brandeis UNIVERSITY

BOSTON UNIVERSITY

Partial Compaction in LSM

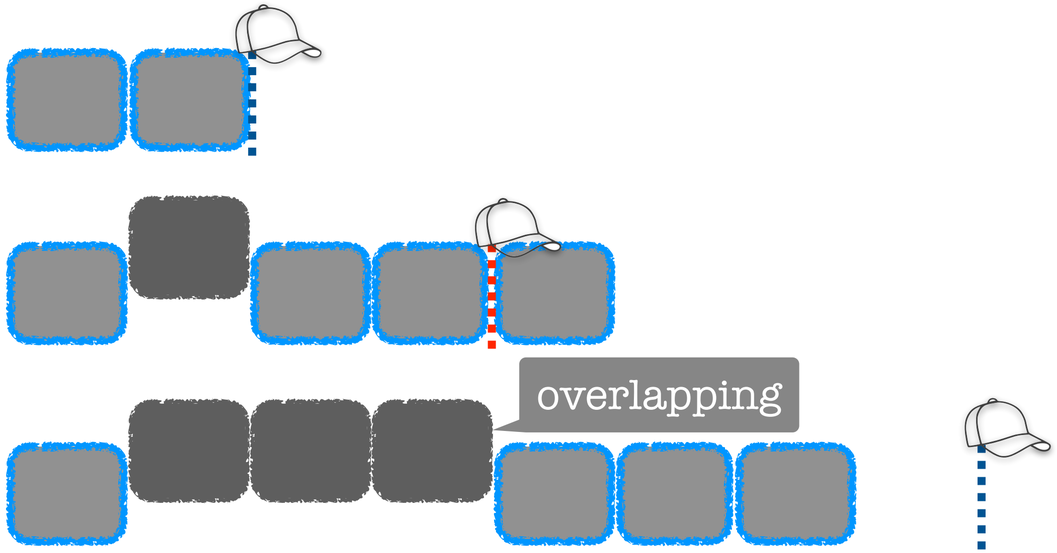


buffer

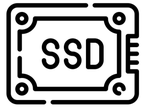
L1

L2

L3



Partial Compaction in LSM



buffer

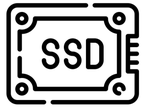
L1

L2

L3

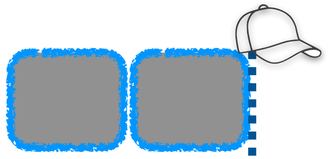
new files

Partial Compaction in LSM



buffer

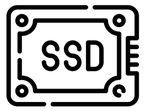
L1



L2

L3

Updates & Deletes in LSM



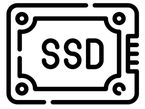
buffer | 5 | 30 | 20 | 81 |

L1

L2 | 10 | 25 | 30 | 55 | 79 | 91 |

L3 | 2 | 5 | 20 | 55 | 65 | 75 | 79 | 95 | 98 | 99 | 101 | 121 | 123 |

Updates & Deletes in LSM



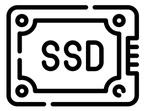
buffer | 5 | 30 | 20 | 81 |

L1

L2 | 10 | 25 | 30 | 55 | 79 | 91 |

L3 | 2 | 5 | 20 | 55 | 65 | 75 | 79 | 95 | 98 | 99 | 101 | 121 | 123 |

Updates & Deletes in LSM



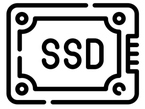
buffer | 5 | 20 | 30 | 81 |

L1

L2 | 10 | 25 | 30 | 55 | 79 | 91 |

L3 | 2 | 5 | 20 | 55 | 65 | 75 | 79 | 95 | 98 | 99 | 101 | 121 | 123 |

Updates & Deletes in LSM



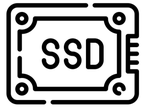
buffer

L1 5 20 30 81 logically invalid

L2 10 25 30 55 79 91

L3 2 5 20 55 65 75 79 95 98 99 101 121 123

Updates & Deletes in LSM



buffer

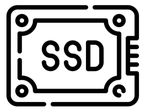
L1 5 20 30 81 logically invalid

L2 10 25 30 55 79 91

L3 2 5 20 55 65 75 79 95 98 99 101 121 123

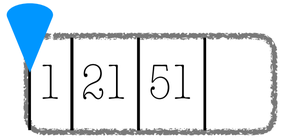
Range Queries in LSM

get(20, 80)

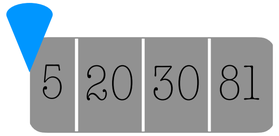


iterator

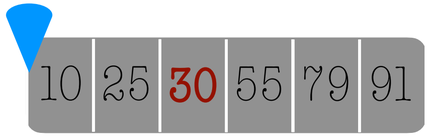
buffer



L1



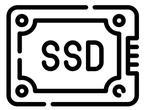
L2



L3

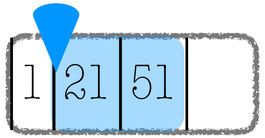
Range Queries in LSM

get(20, 80)

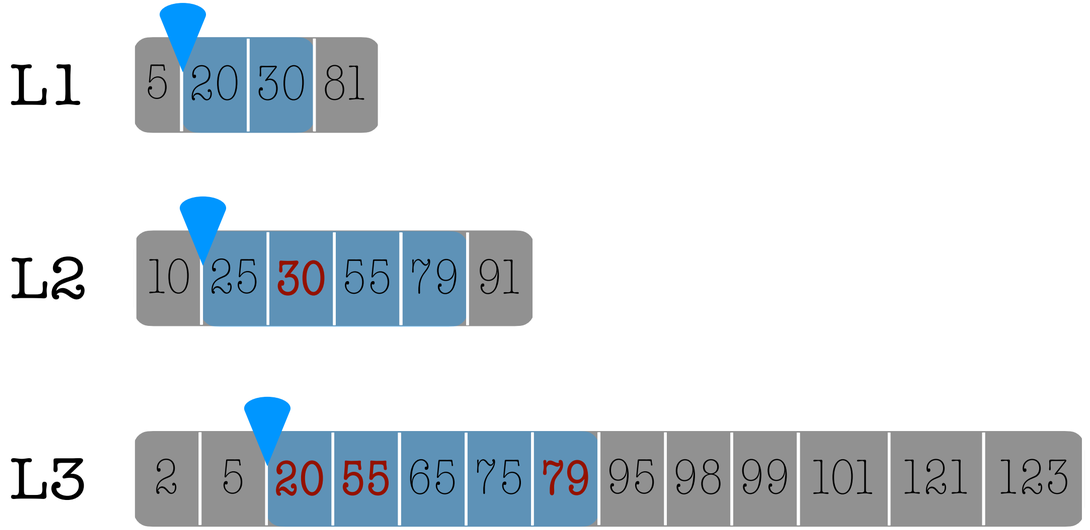


iterator

buffer



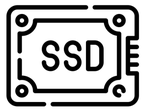
response



K-way merge

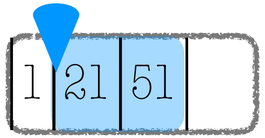
Range Queries in LSM

get(20, 80)

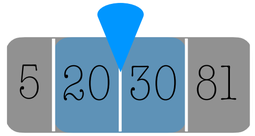


iterator

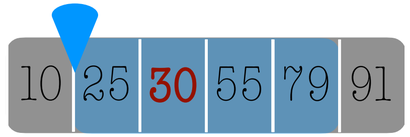
buffer



L1



L2



L3

response

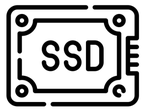
20

~~20~~

K-way merge

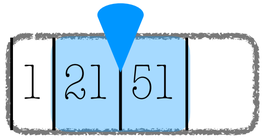
Range Queries in LSM

get(20, 80)

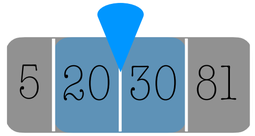


iterator

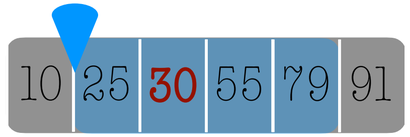
buffer



L1



L2



L3

response

20 21

~~20~~

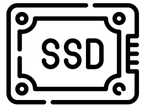
K-way merge

Brandeis UNIVERSITY

BOSTON UNIVERSITY

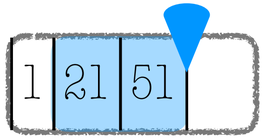
Range Queries in LSM

get(20, 80)

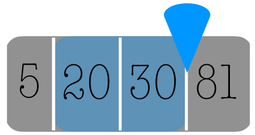


iterator

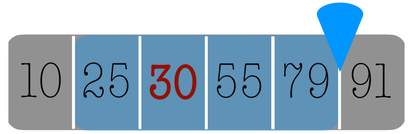
buffer



L1



L2



L3

response

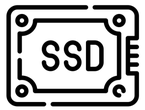
20 21 25 30 51 55 65 75 79

~~20~~ ~~30~~ ~~55~~ ~~79~~

K-way merge

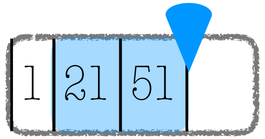
Range Queries in LSM

get(20, 80)



iterator

buffer

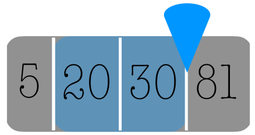


response

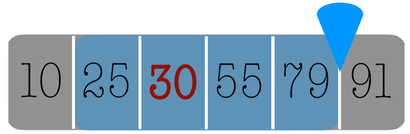
20 21 25 30 51 55 65 75 79

~~20~~ ~~30~~ ~~55~~ ~~79~~

L1



L2



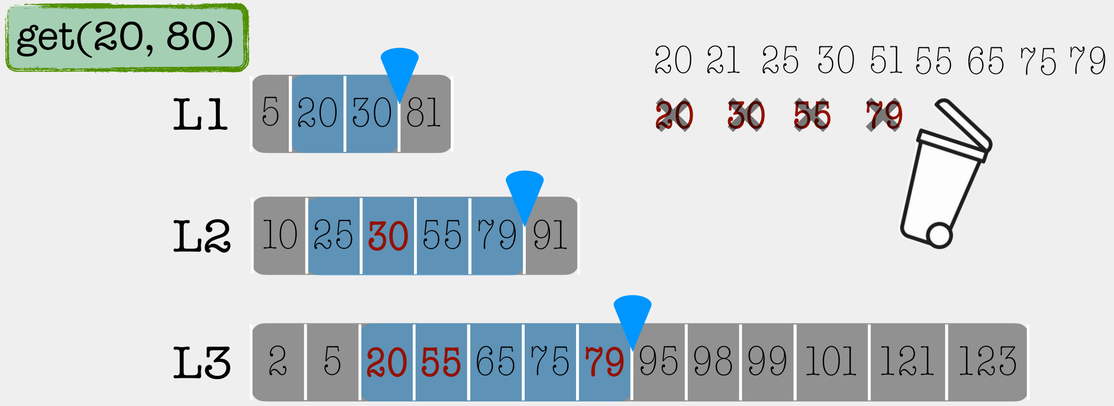
L3

repetitive merging

wasted work

Limitations of the State-of-the-Art

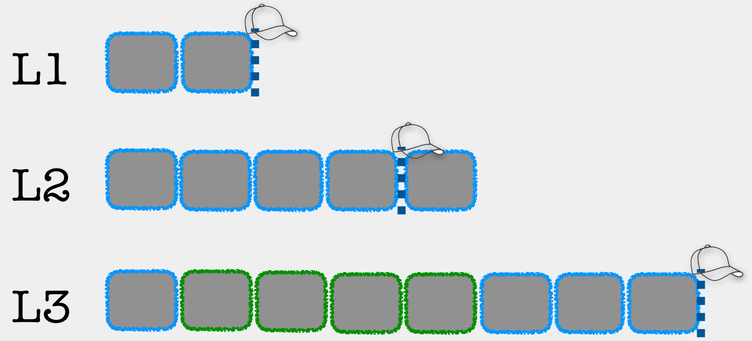
Problem 1



repetitive merging

wasted work

Problem 2

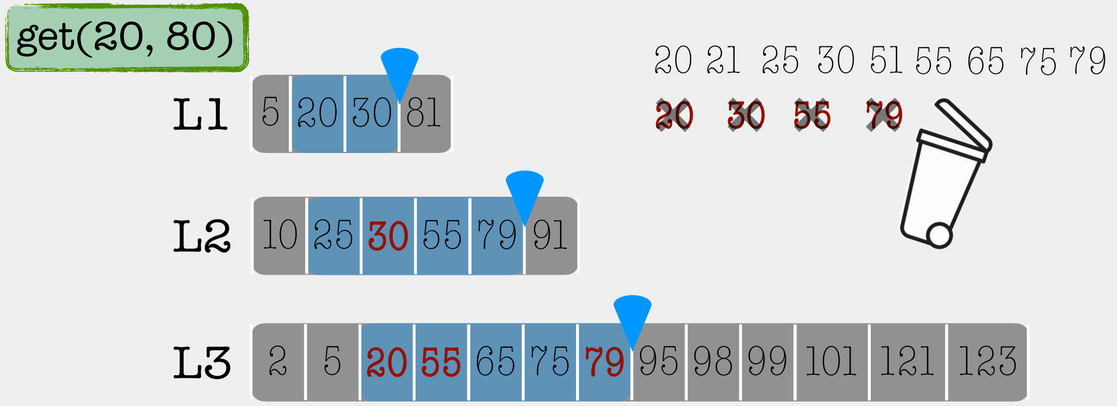


Brandeis UNIVERSITY

BOSTON UNIVERSITY

Limitations of the State-of-the-Art

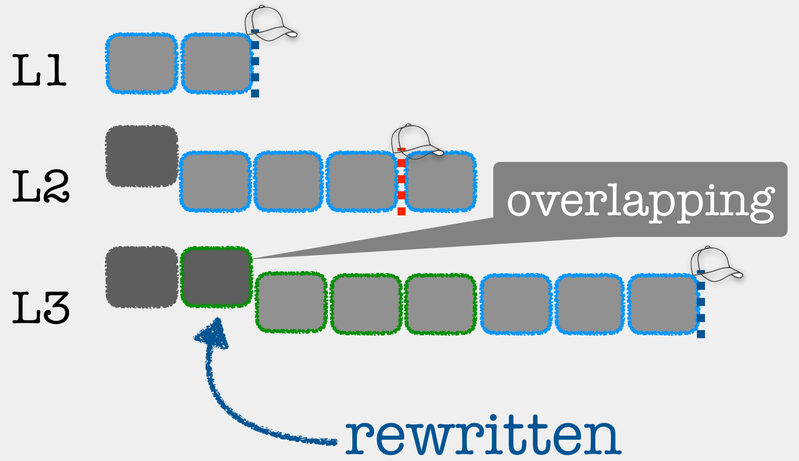
Problem 1



repetitive merging

wasted work

Problem 2

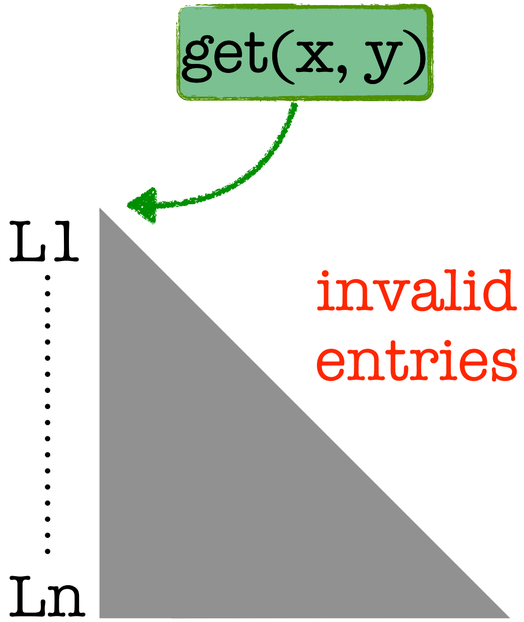


read amp.

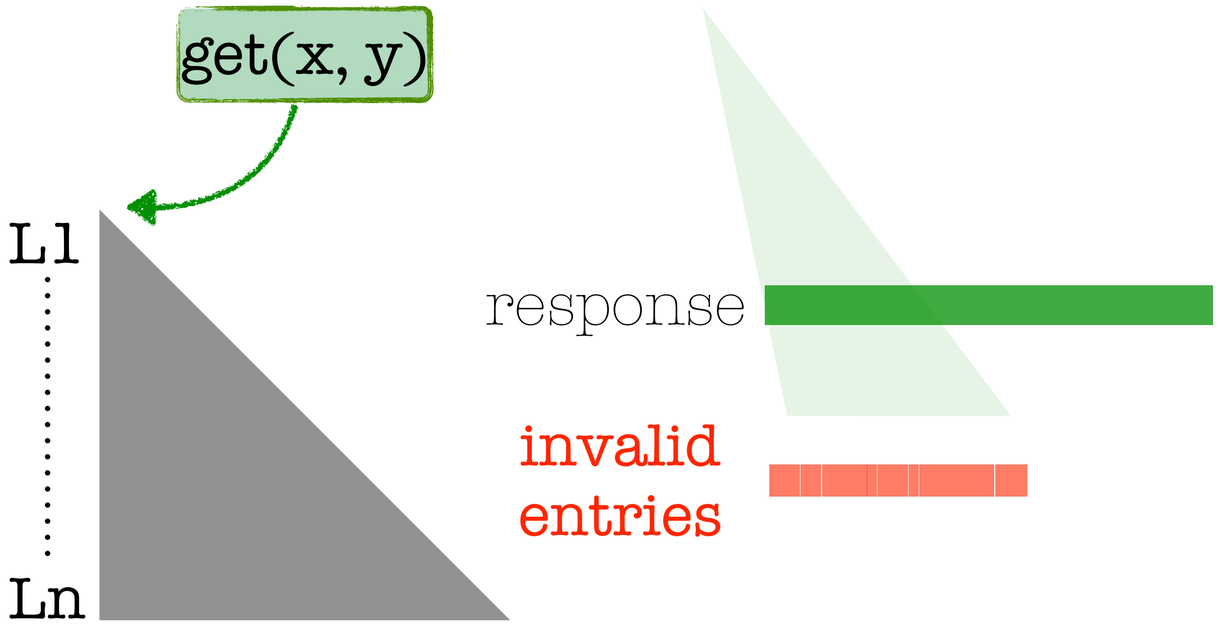
write amp.

Can we do something better?

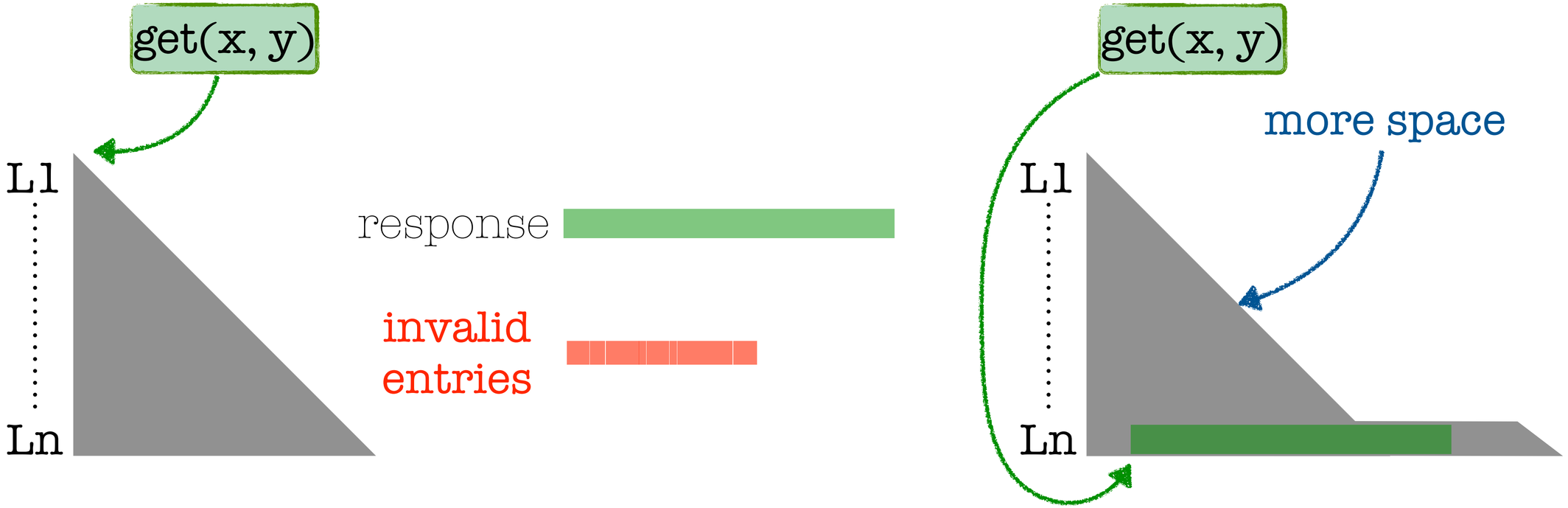
Naive Solution: Merge-on-Scan



Naive Solution: Merge-on-Scan



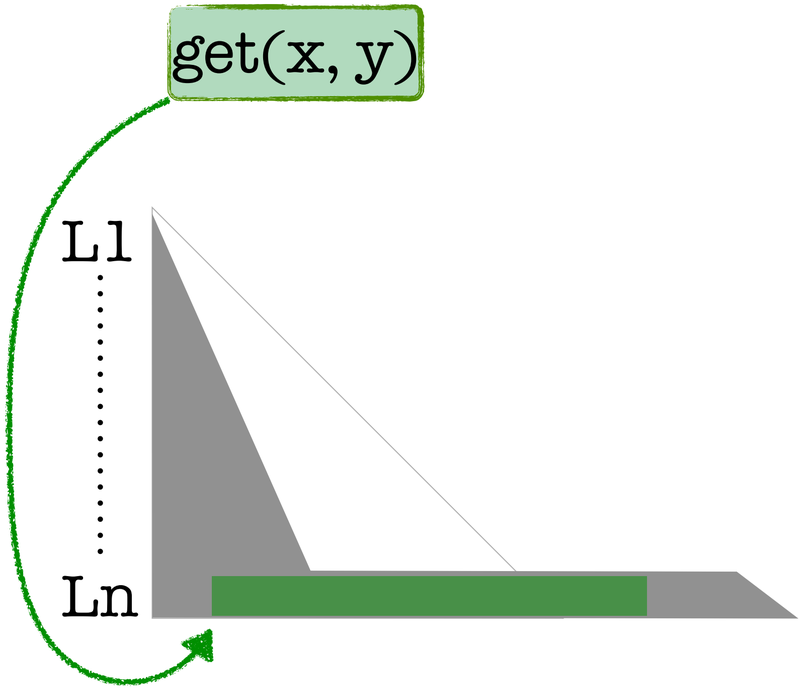
Naive Solution: Merge-on-Scan



less read & write amp.

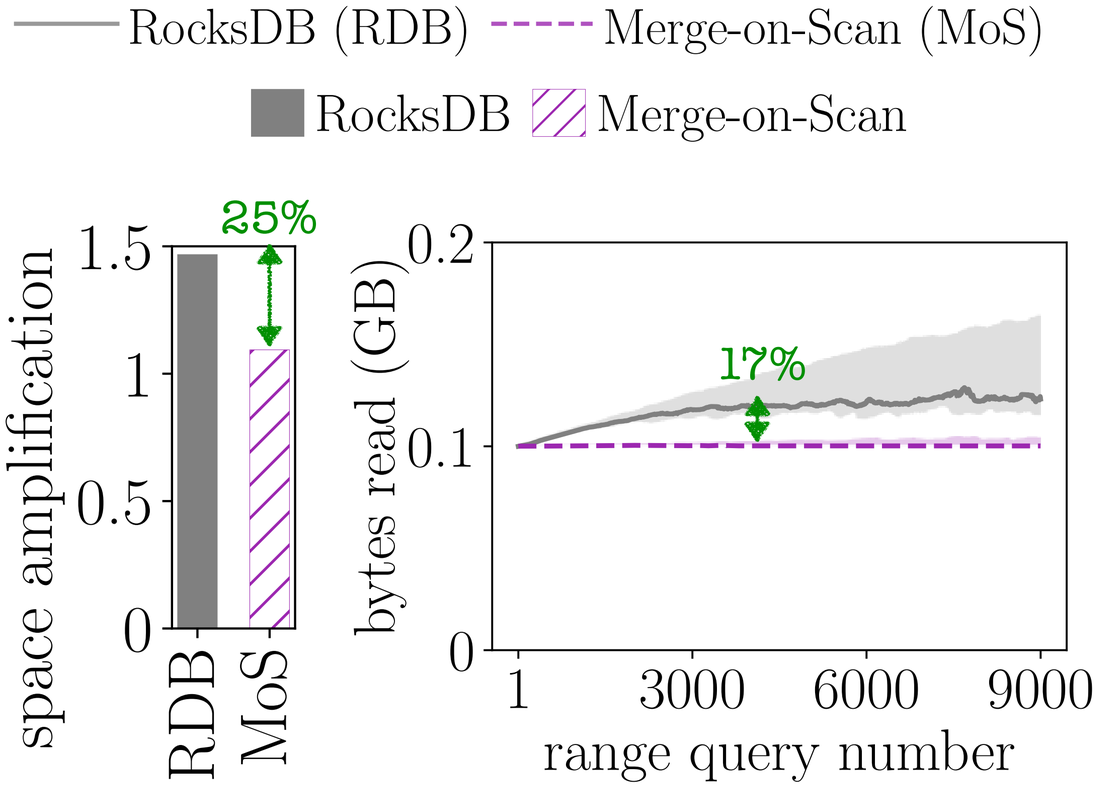
no repetitive merging

Merge-on-Scan



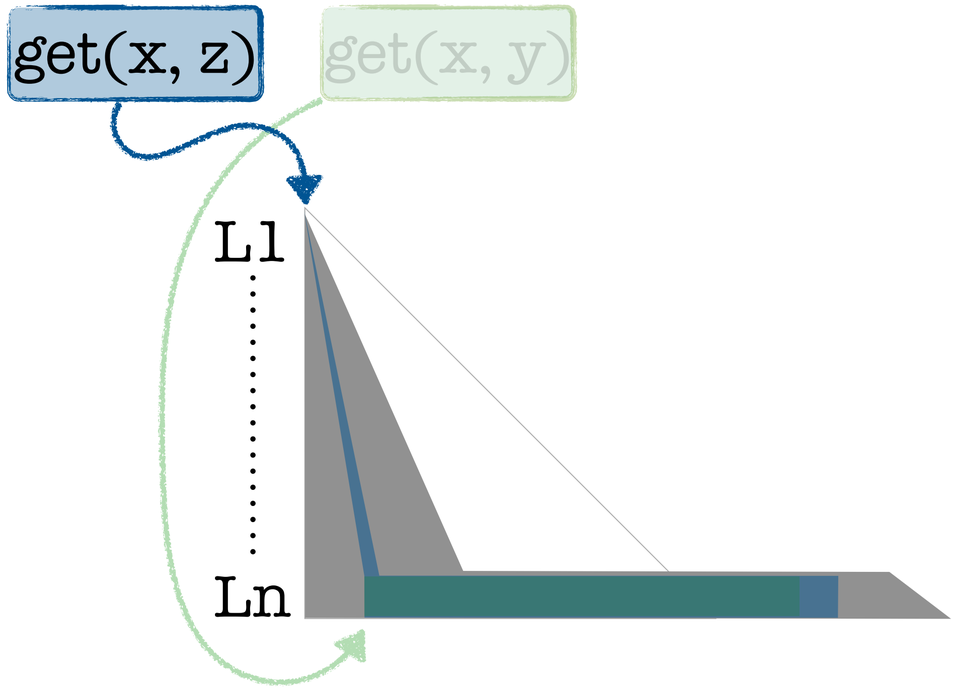
less read & write amp.

no repetitive merging



MoS reduces space amplification by 25% and reads up to 17% less data during range queries

Merge-on-Scan



slower RQ

increases write amp.

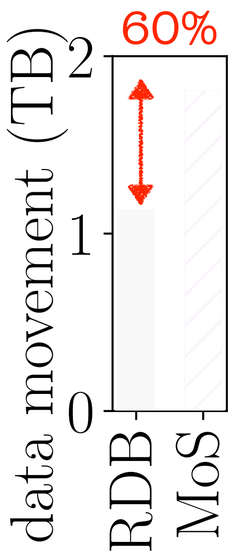
M: buffer size
E: entry size
T: size ratio

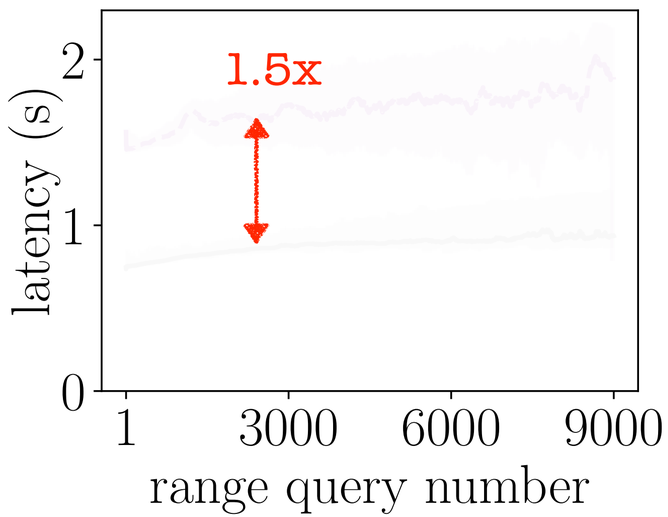
Merge-on-Scan

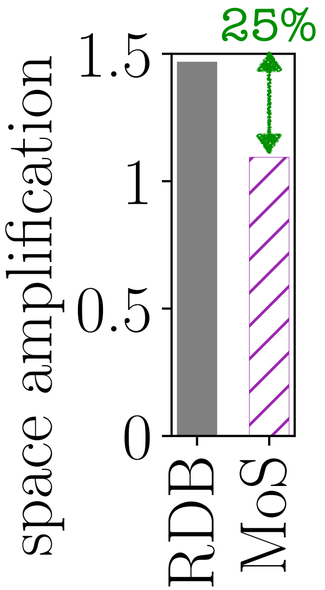
I: inserts
U: updates
S: range queries

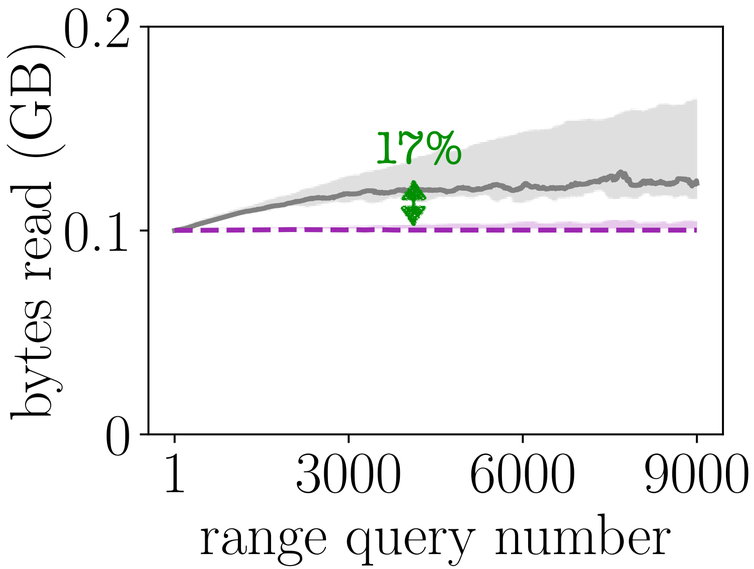
M=4MB E=128B T=6 I=8.4M U=8.4M S=9K

RocksDB (RDB) Merge-on-Scan (MoS) RocksDB Merge-on-Scan









MoS is moving (reading & writing) more data overall

range queries are 1.5x slower than RocksDB

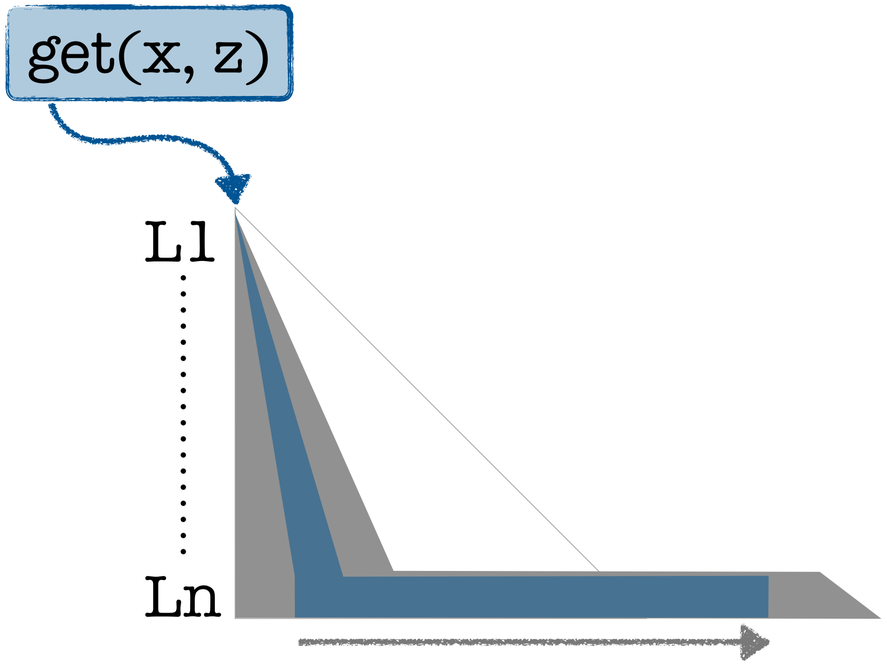
MoS reduces space amplification by 25% and reads up to 17% less data during range queries

Brandeis UNIVERSITY

BOSTON UNIVERSITY

DiSC lab

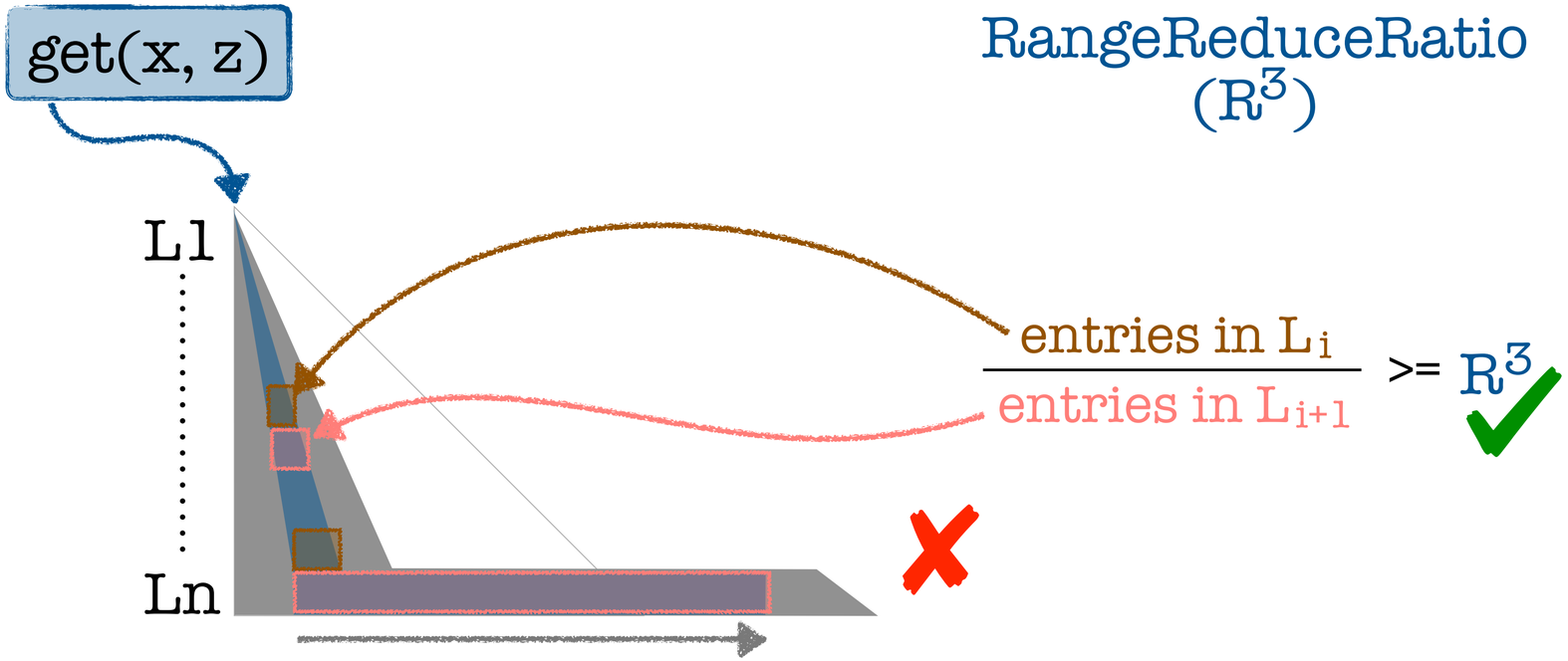
Bounded-Merge



slower RQ

increases write amp.

Bounded-Merge



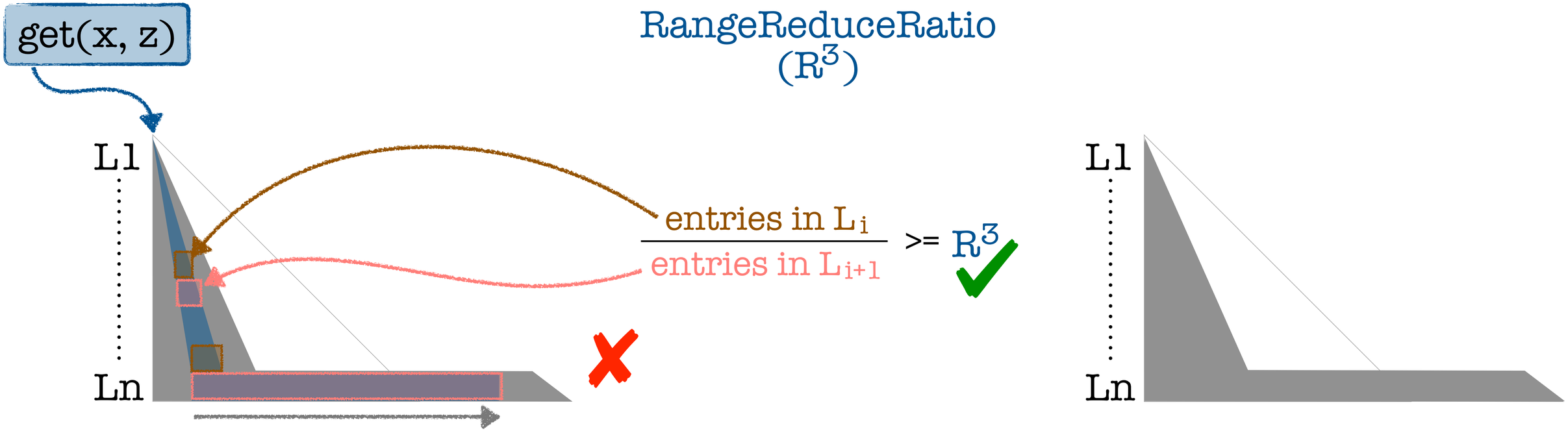
RangeReduceRatio
(R^3)

$$\frac{\text{entries in } L_i}{\text{entries in } L_{i+1}} >= R^3$$

slower RQ

increases write amp.

Bounded-Merge



slower RQ

increases write amp.

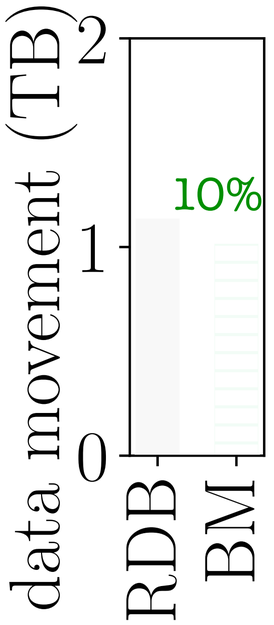
M: buffer size
E: entry size
T: size ratio

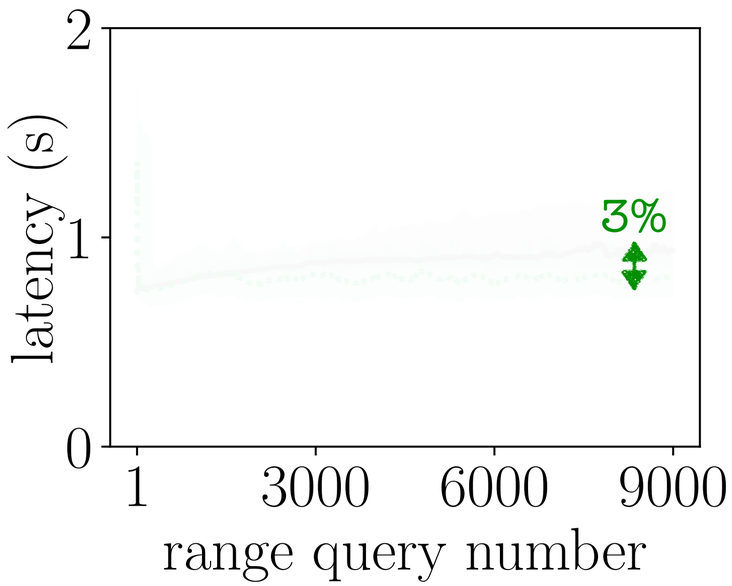
Bounded-Merge

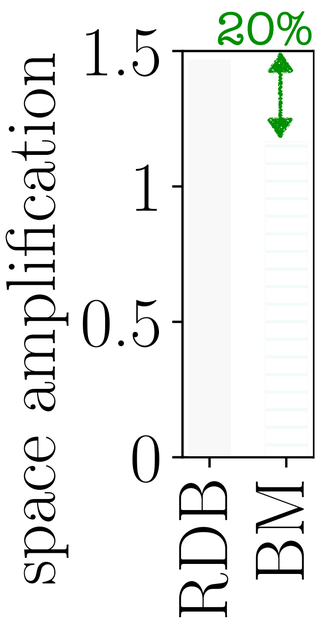
I: inserts
U: updates
S: range queries

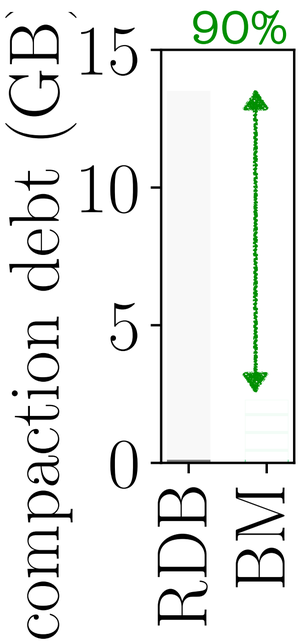
M=4MB E=128B T=6 I=8.4M U=8.4M S=9K s=0.1

RocksDB (RDB) Bounded-Merge (BM) RocksDB Bounded-Merge









BM is moving **10% less data** than RocksDB

range queries are 3% faster in BM than RocksDB

BM offers 20% better space amp. & 90% less compaction debt

Brandeis UNIVERSITY

BOSTON UNIVERSITY

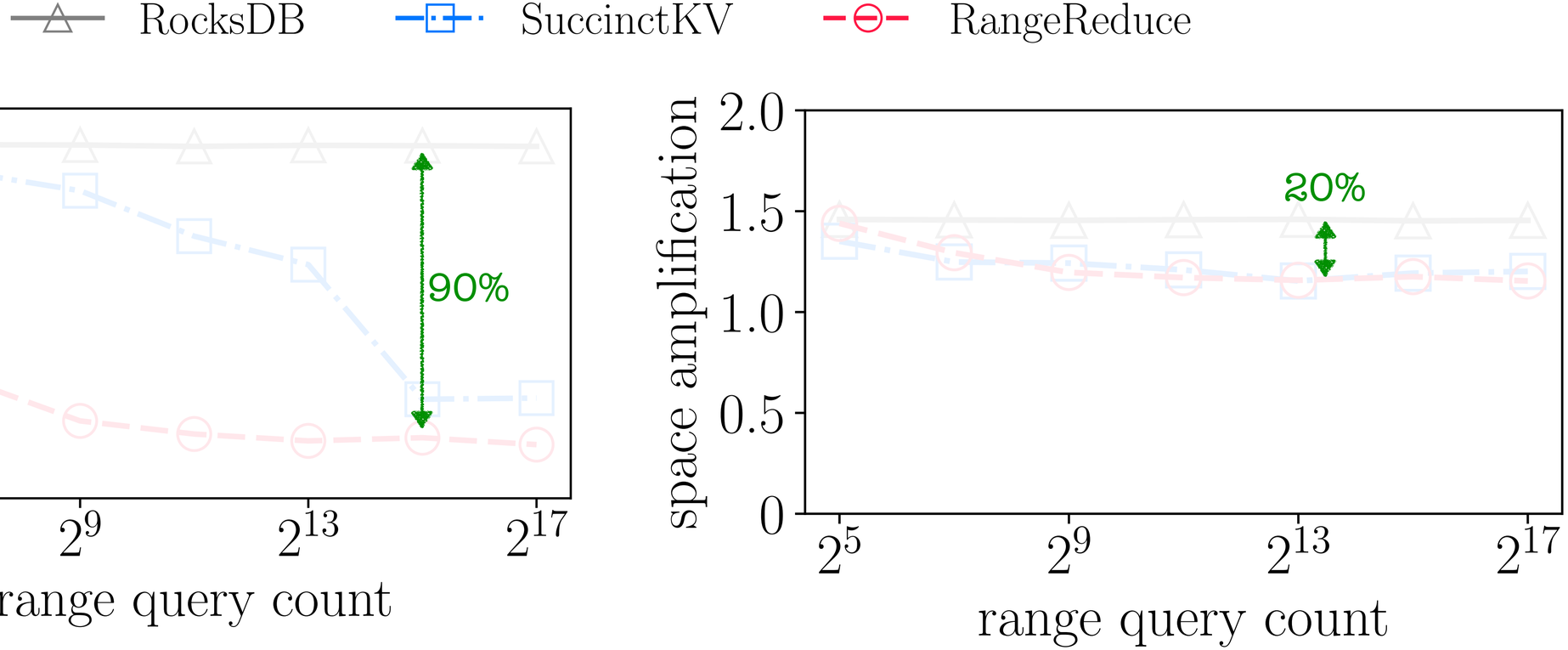
DiSC lab

M: buffer size
E: entry size
T: size ratio

Experimental Evaluation

I: inserts
U: updates
S: range queries

M=4MB E=128B T=6 I=8.4M S+U=8.4M s=0.1



Reduces the future work of compaction by up to 90%

Improves the space amp. by up to **20%** than RocksDB

Brandeis UNIVERSITY

BOSTON UNIVERSITY

RangeReduce: Query-Driven LSM Compactions

LSMs perform lots of repetitive work for the RQ-heavy workloads

Compactions reduce the logically invalid key, but not to the full potential

RQs compact data between a requested range, but it is thrown away

RangeReduce leverages RQs to reduce future work, improving the performance

Shubham Kaushik
https://shubhamkaushik.com

Manos Athanassoulis

Subhadeep Sarkar

Brandeis UNIVERSITY

BOSTON UNIVERSITY