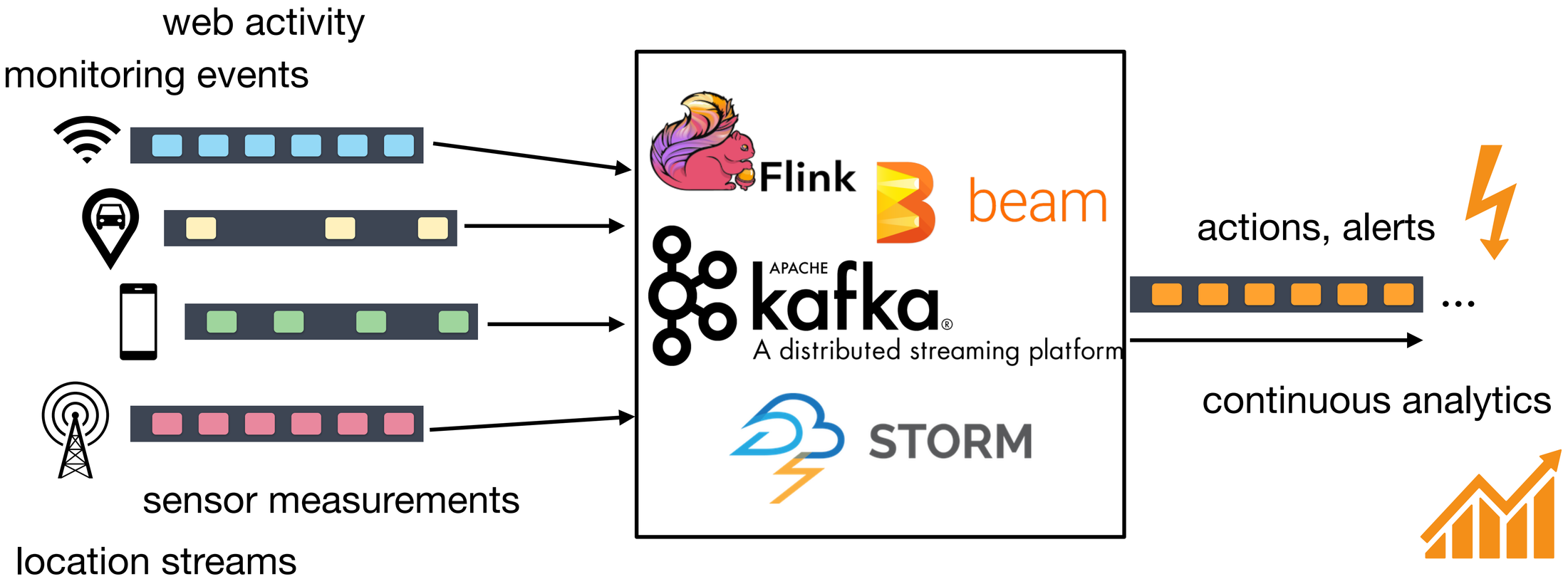
BOSTON UNIVERSITY

A New Benchmark Harness for Systematic and Robust Evaluation of Streaming State Stores

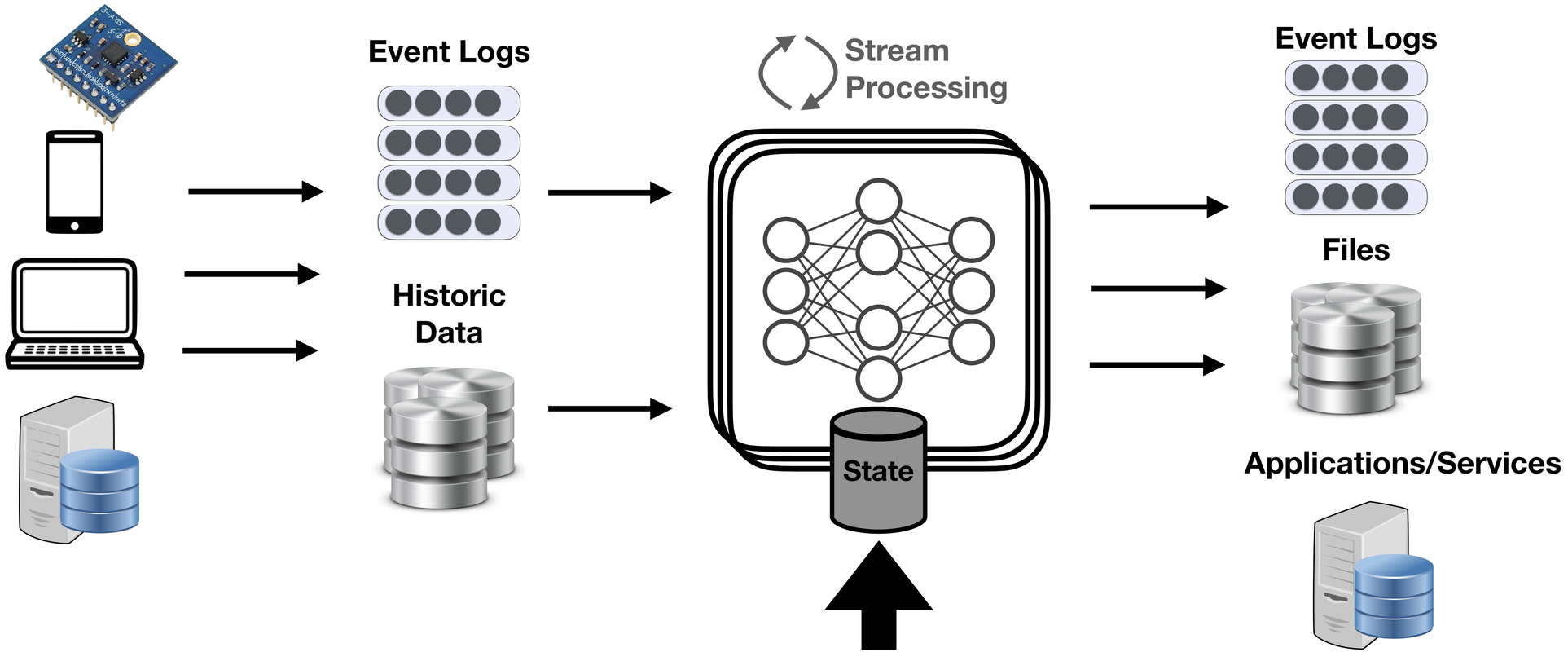
Showan Asyabi, Yuanli Wang, John Liagouris, **Vasiliki Kalavri**, Azer Bestavros

https://sites.bu.edu/casp

Data stream processing systems enable low-latency decision making and continuous analytics



Any non-trivial streaming computation accumulates and maintains state

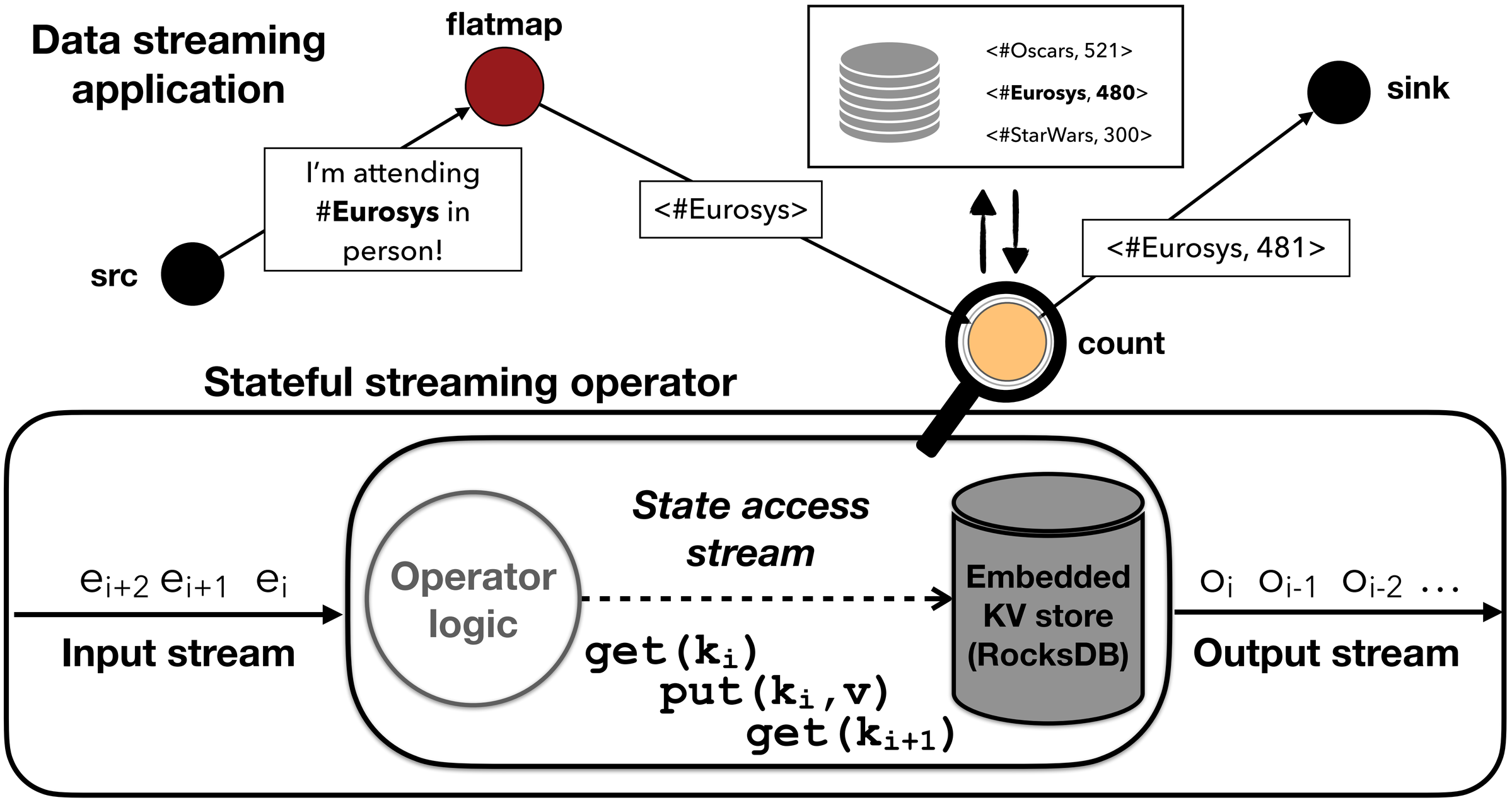


window contents

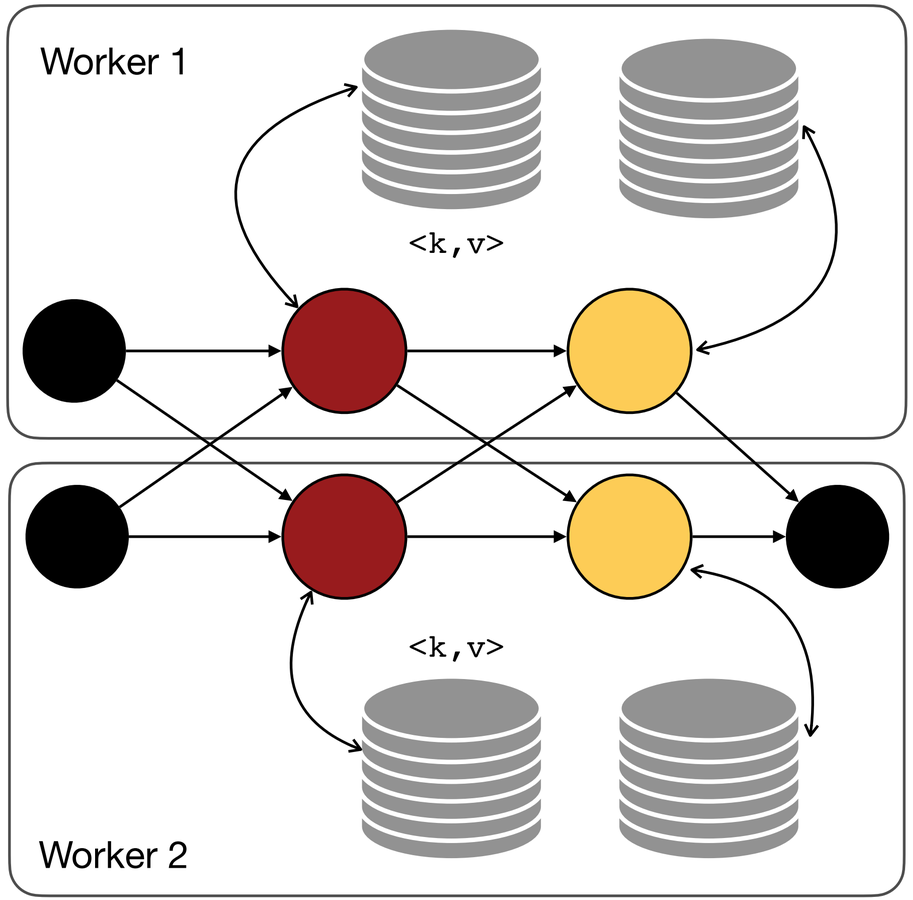
input offsets

rolling aggregations

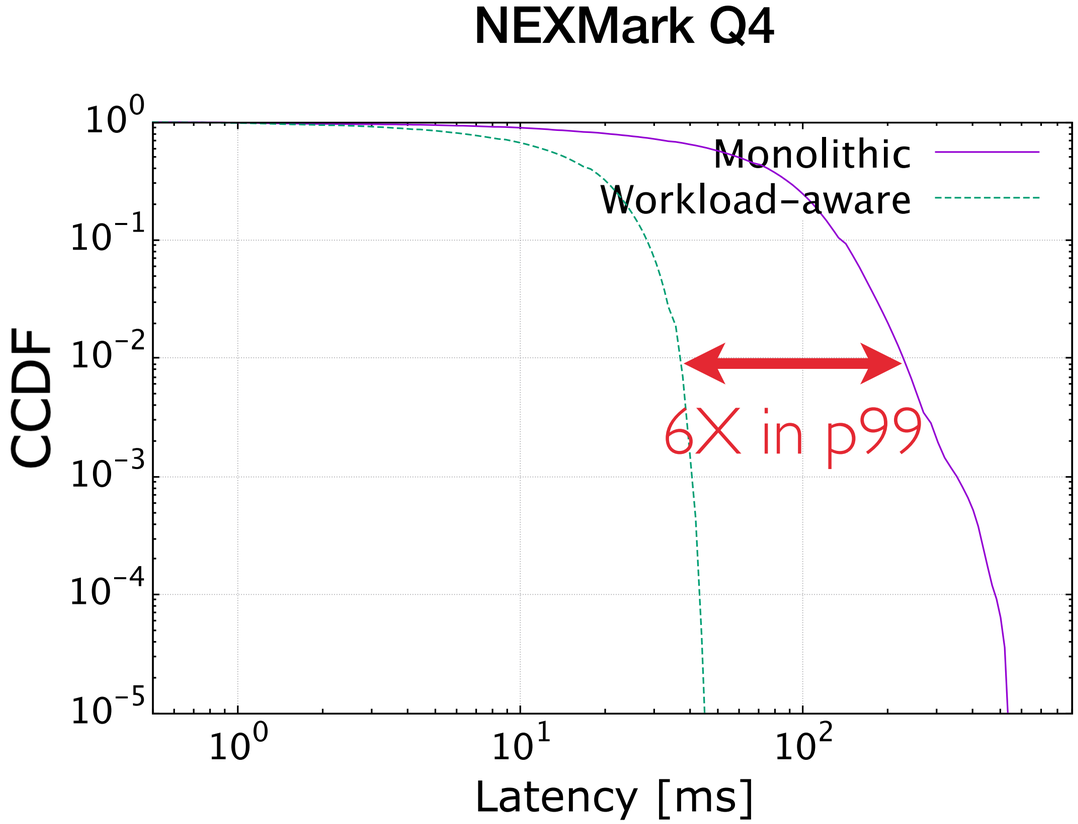
ML models


Data streaming application
flatmap
I'm attending #Eurosys in person!
src
<#Eurosys>
<#Oscars, 521>
<#Eurosys, 480>
<#StarWars, 300>
<#Eurosys, 481>
sink
count
Stateful streaming operator
Operator logic
State access stream
get(ki)
put(ki,v)
get(ki+1)
Embedded KV store (RocksDB)
ei+2 ei+1 ei
Input stream
oi oi-1 oi-2 ...
Output stream

Current practice: Monolithic state management



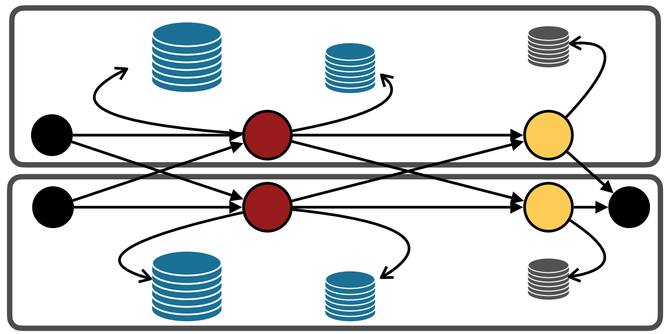
One key-value store (**RocksDB**) per stateful operator instance



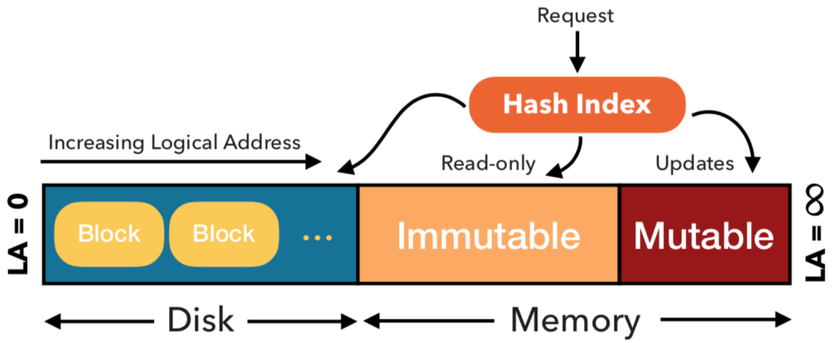
In support of workload-aware streaming state management
(Vasiliki Kalavri, John Liagouris, HotStorage'20)

Alternative designs might improve performance but are cumbersome to evaluate

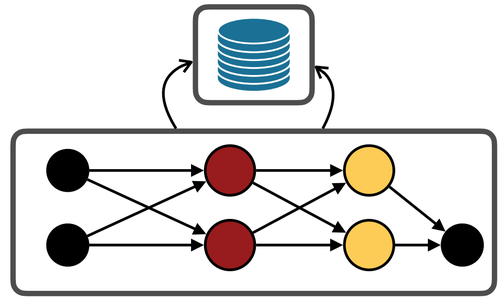
Different store configuration per operator



Replace KV Store



Shared external KV store

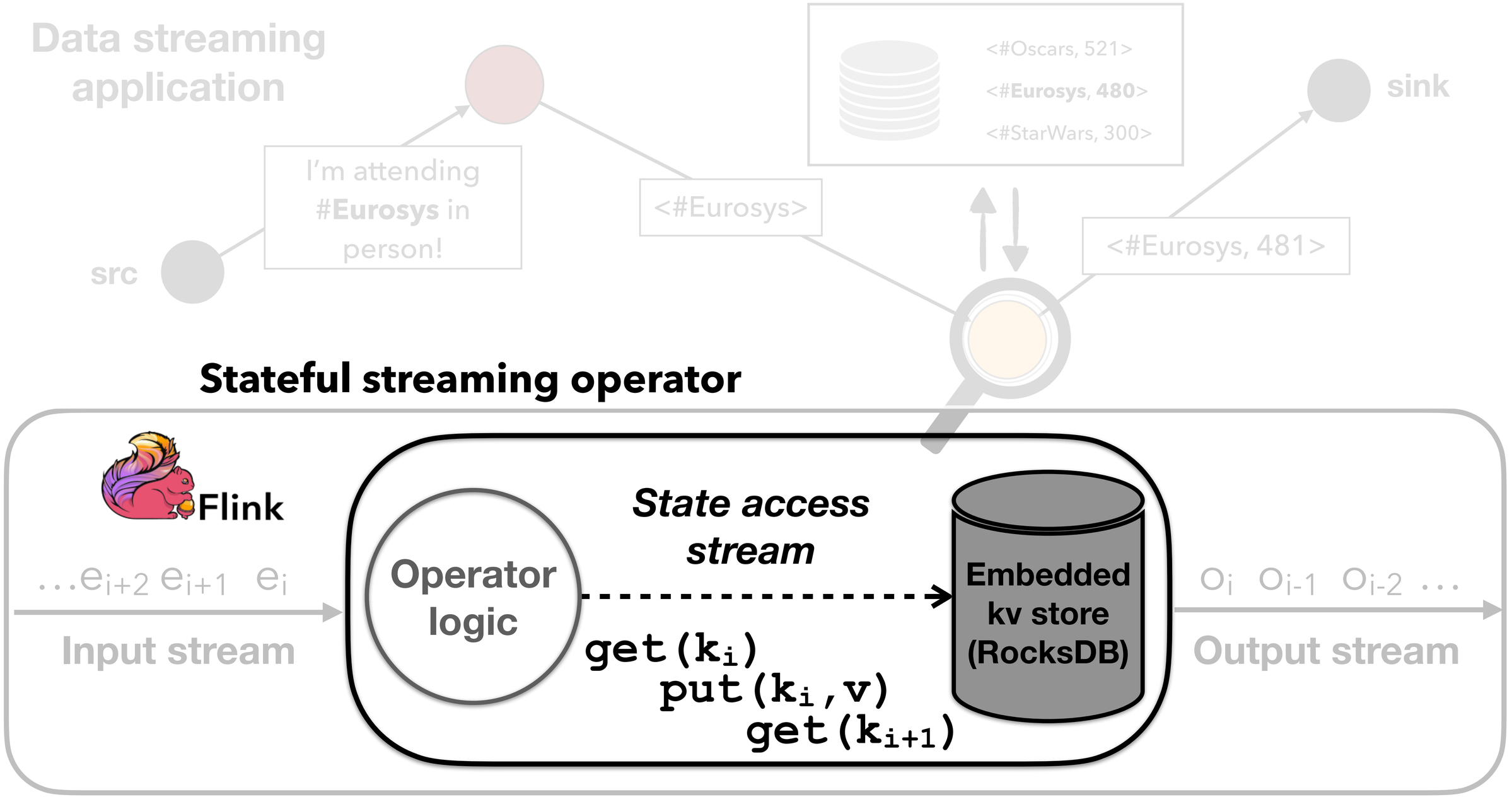


- Integrate new KV stores with a streaming engine
- Configure and deploy a cluster
- Instrument the streaming engine to collect traces

Our contributions

- We report on the first **empirical characterization study** of streaming state access workloads using 3 real-world data streams.
- We empirically confirm that **YCSB workloads cannot approximate** state access traces produced by streaming operators.
- We design and implement ***Gadget*, a new benchmark harness** for **systematic** and **robust** evaluation of standalone streaming state stores.
- We perform a **performance evaluation study of four KV stores** for streaming state management.

Characterizing streaming state access workloads


Data streaming application
I'm attending #Eurosys in person!
src
<#Eurosys>
<#Oscars, 521>
<#Eurosys, 480>
<#StarWars, 300>
<#Eurosys, 481>
sink
Stateful streaming operator
Flink
...e$_{i+2}$ e$_{i+1}$ e$_i$
Input stream
Operator logic
State access stream
get(k$_i$)
put(k$_i$,v)
get(k$_{i+1}$)
Embedded kv store (RocksDB)
o$_i$ o$_{i-1}$ o$_{i-2}$ ...
Output stream

Workload composition

	Borg				Azure			
	GET	PUT	MERGE	DELETE	GET	PUT	MERGE	DELETE
Tumbl-Incr	**0.5**	0.459	0	0.041	**0.5**	0.405	0	0.095
Sliding-Incr	**0.5**	0.459	0	0.041	**0.5**	0.405	0	0.095
Session-Incr	**0.575**	0.281	0.062		**0.544**	0.202	0.064	0.189
Tumbl-Hol	0.076	0	**0.847**		0.165	0	**0.669**	0.165
Sliding-Hol	0.076	0	**0.847**		0.159	0	**0.681**	0.159
Session-Hol	0.409	0	**0.477**		**0.429**	0	0.334	0.238
Join-Cont	**0.59**	0.006	0.39	0.013	-	-	-	-
Join-Interval	**0.446**	**0.446**	0	0.108	-	-	-	-
Aggregation	**0.5**	**0.5**	0	0	**0.5**	**0.5**	0	0

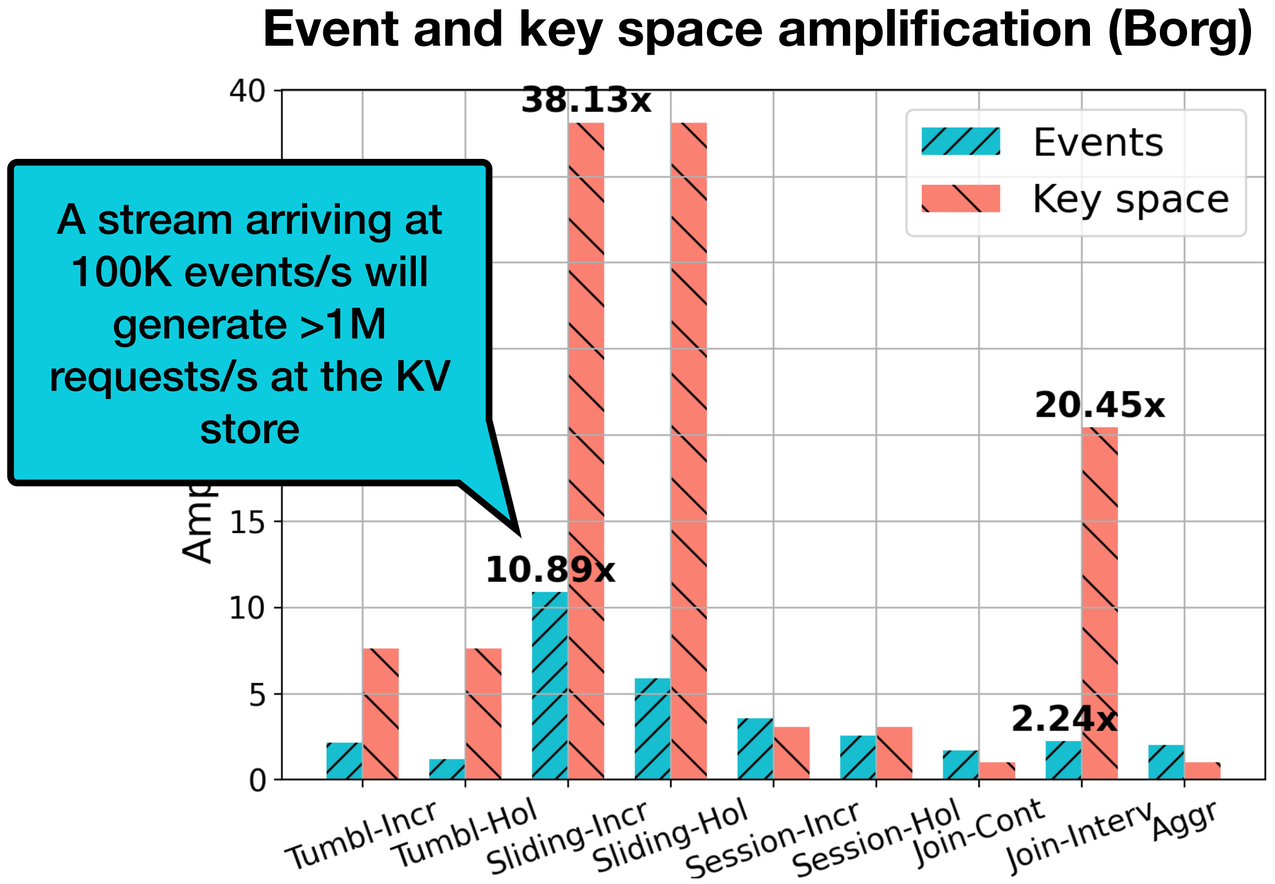
Composition varies across input streams

Composition varies across operators

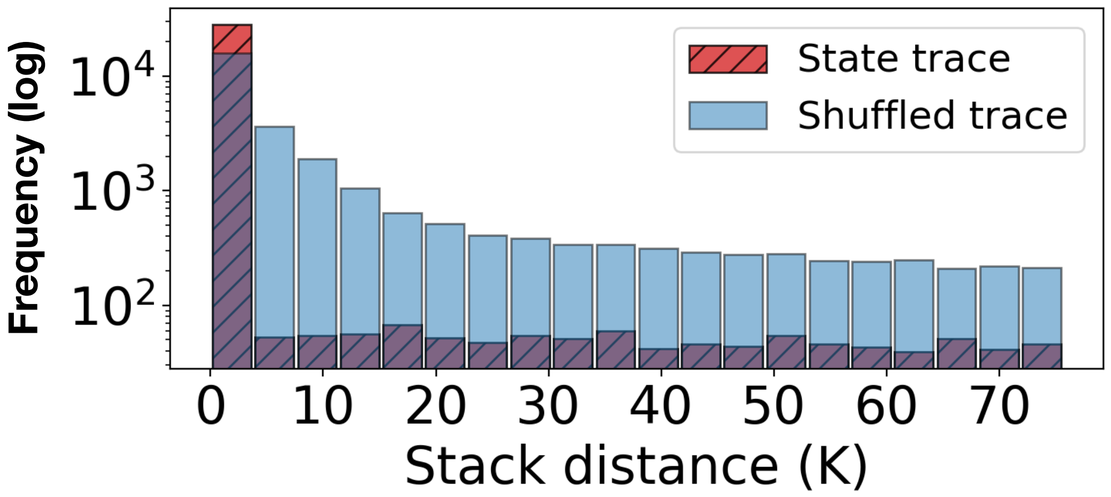
Windows and joins make frequent deletes

The state access stream differs significantly from the input stream

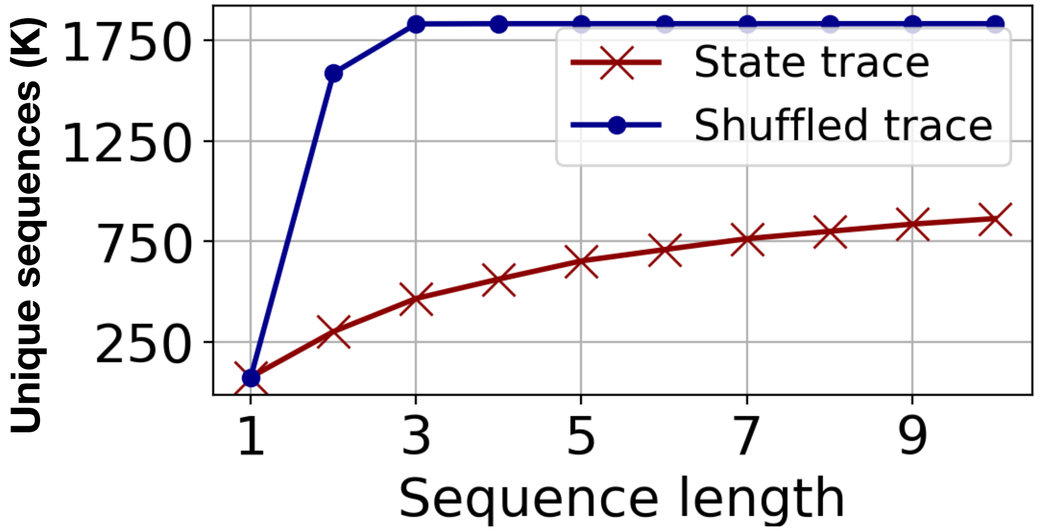
- The state store accepts a much higher request load than the stream arrival rate.
- Most workloads exhibit key distributions different from those of their respective input streams.



State access streams exhibit high temporal and spatial locality



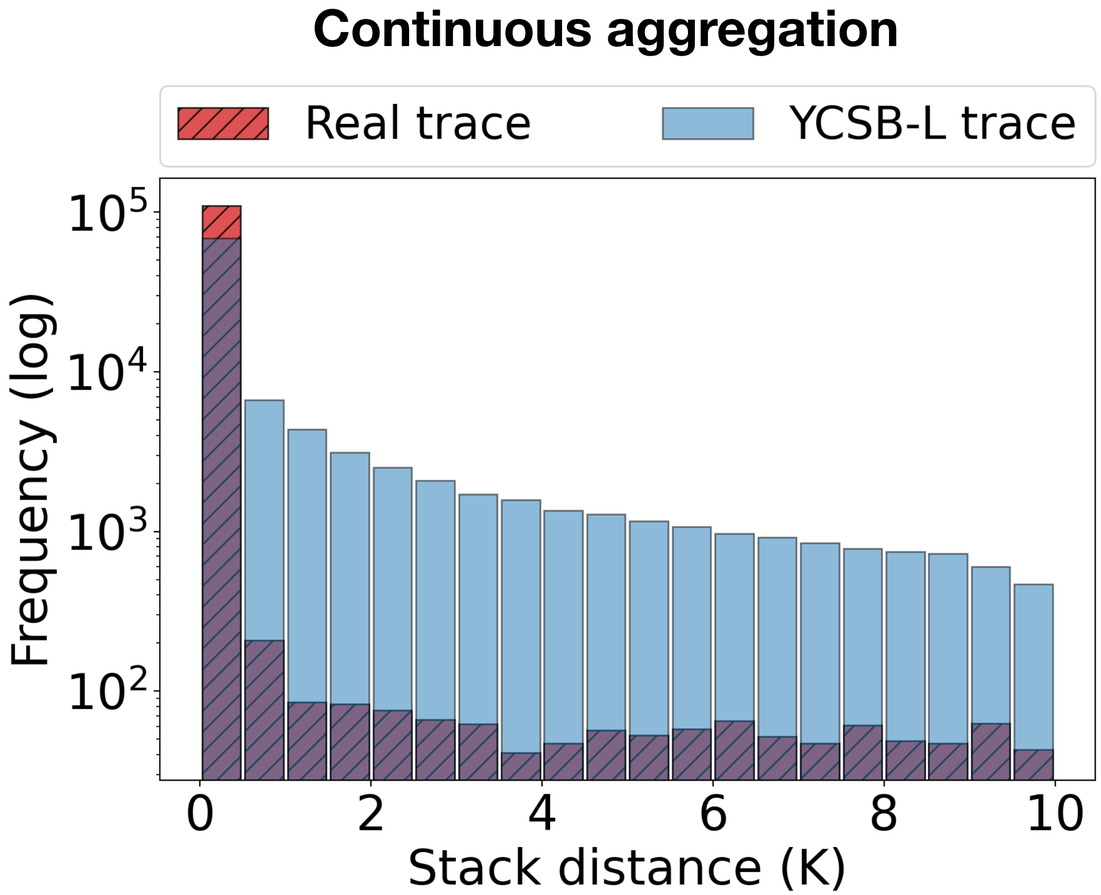
Stack distance: distribution of unique keys accessed between consecutive operations on the same key.



Unique sequences occurring in a state access stream with a maximum length of up to 10.

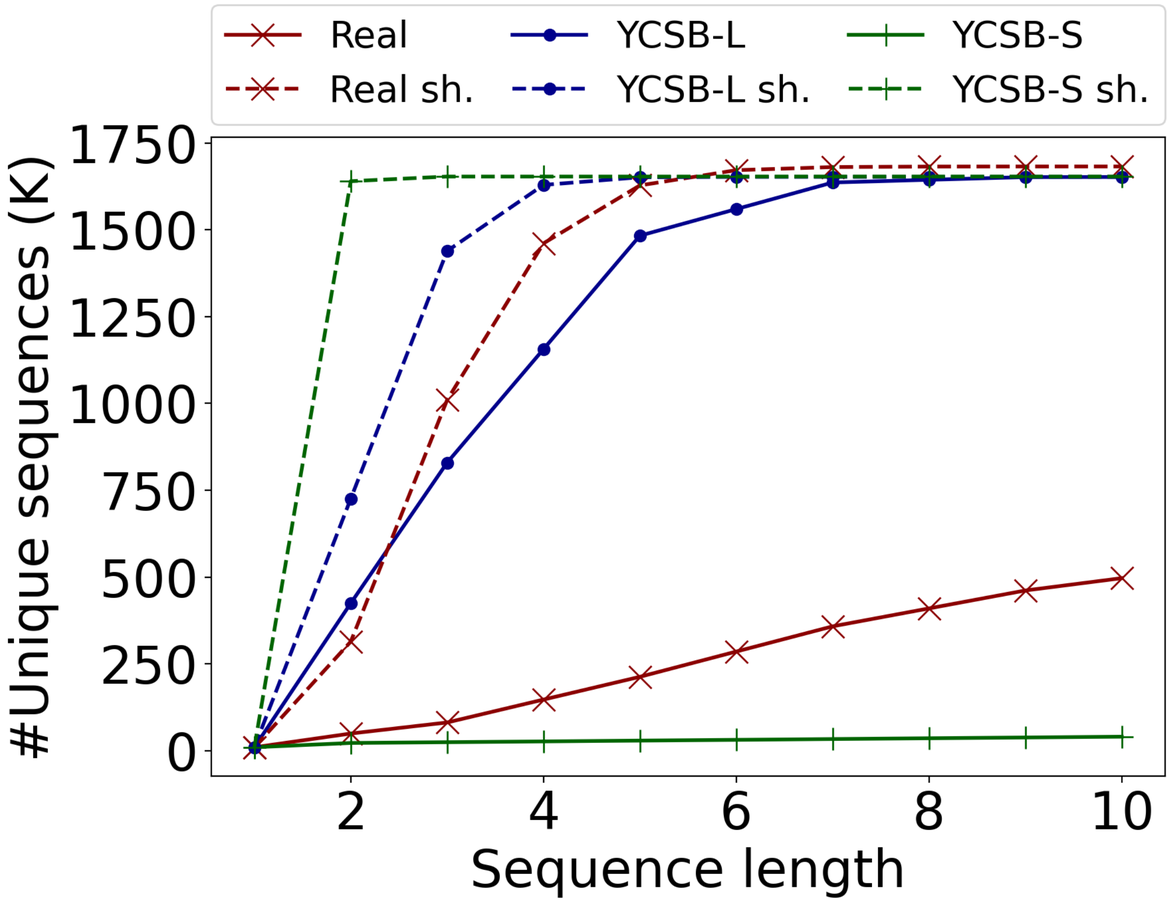
Can we approximate streaming state access workloads with YCSB?

The highest locality achieved by YCSB is much lower than that exhibited in real traces.



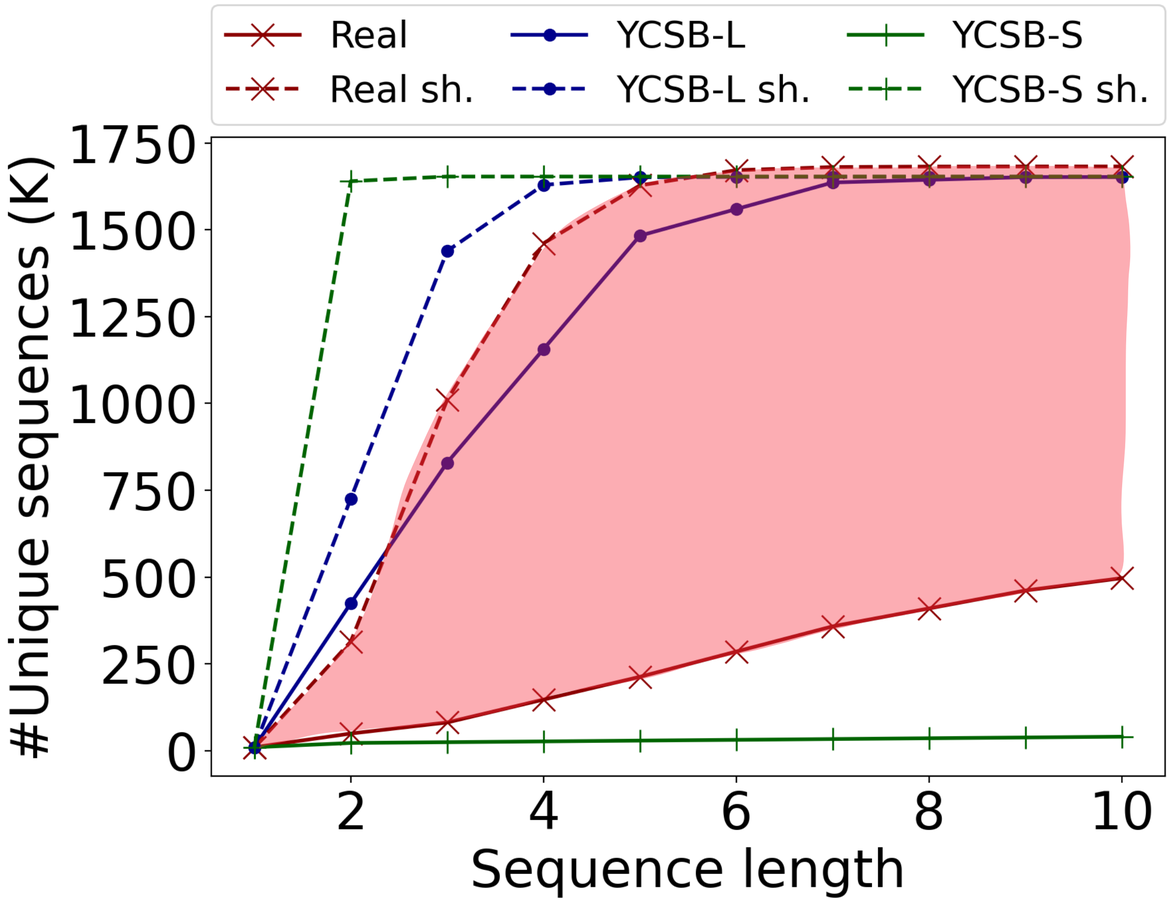
- YCSB-L: `latest`

YCSB workloads **exhibit either temporal or spatial locality** but not both.



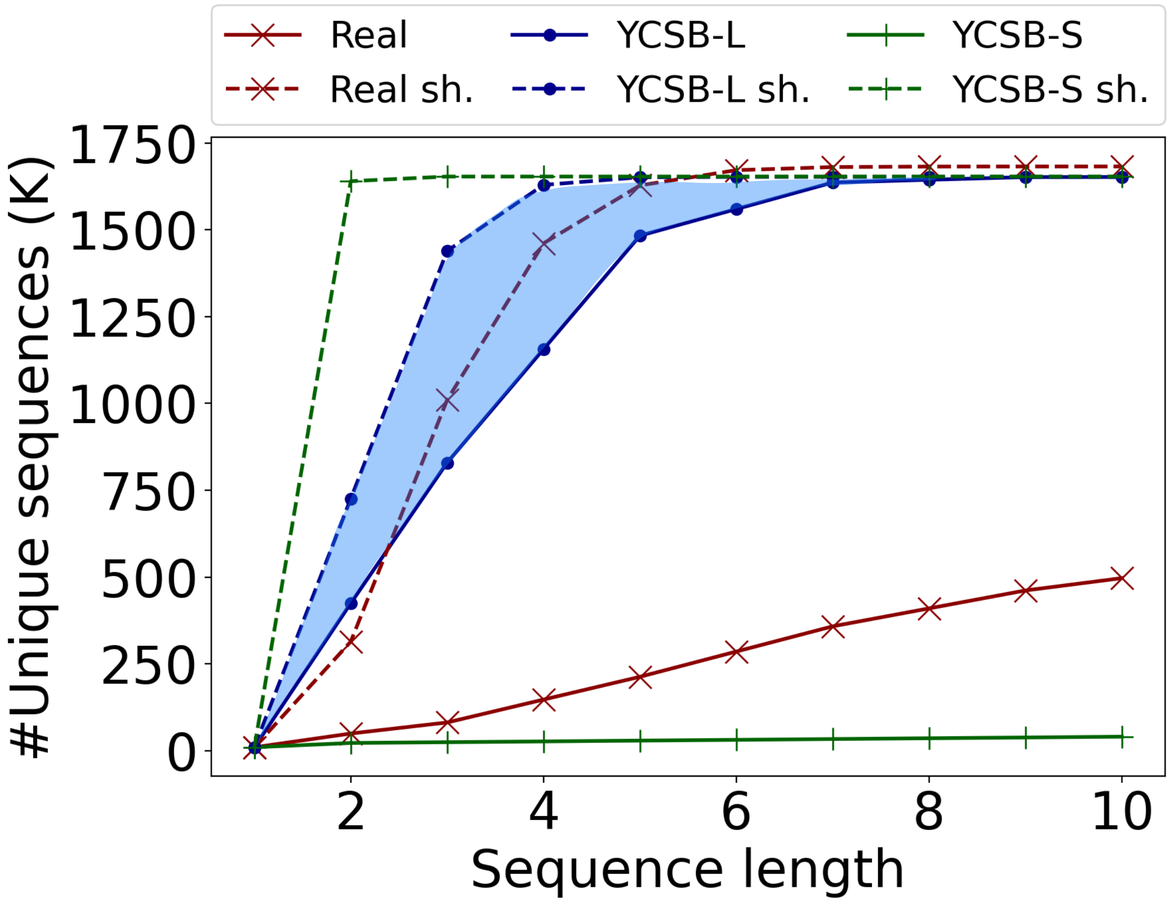
- YCSB-L: `latest`
- YCSB-L sh.: `latest` shuffled
- YCSB-S: `sequential`
- YCSB-S: `sequential` shuffled

YCSB workloads **exhibit either temporal or spatial locality** but not both.



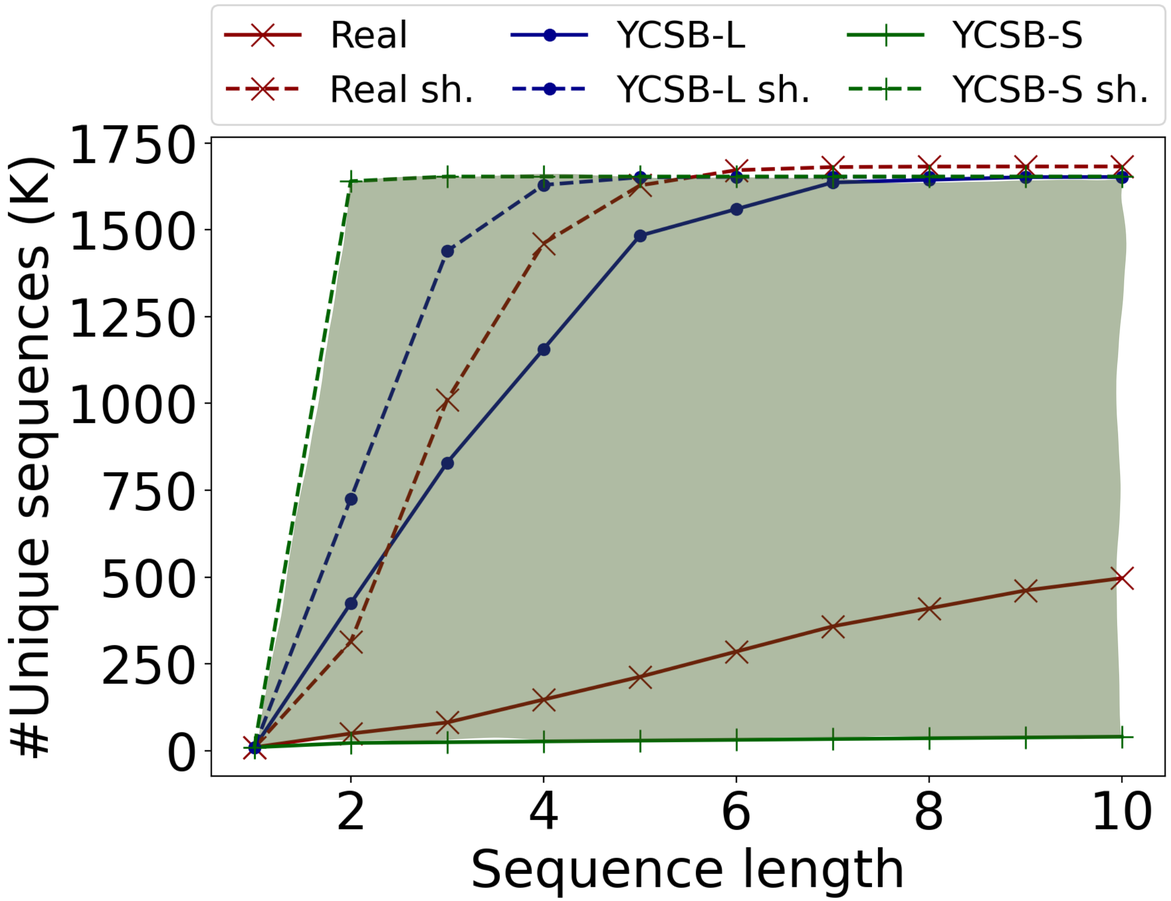
- YCSB-L: `latest`
- YCSB-L sh.: `latest` shuffled
- YCSB-S: `sequential`
- YCSB-S: `sequential` shuffled

YCSB workloads **exhibit either temporal or spatial locality** but not both.



- YCSB-L: `latest`
- YCSB-L sh.: `latest` shuffled
- YCSB-S: `sequential`
- YCSB-S: `sequential` shuffled

YCSB workloads **exhibit either temporal or spatial locality** but not both.



- YCSB-L: `latest`
- YCSB-L sh.: `latest` shuffled
- YCSB-S: `sequential`
- YCSB-S: `sequential` shuffled

The Gadget Benchmark Harness

Gadget enables systematic and robust evaluation of alternative designs

A new benchmark harness that:

- Generates **representative** workloads by closely simulating the state access logic of streaming operators
- Achieves **high accuracy** by exposing a set of configurable parameters
  - arrival rate distribution
  - event time skew
  - watermark frequency
- Provides **eleven predefined workloads**, supports custom operator implementation, and offers connectors to **four KV stores**

Gadget input

operator.type = **tumbling.incremental**
window.length = **5**
expected.num.operations = **10000000**
wrapper.type = **rocksdb**
operator.key.size.distrib.params.constant = **10**
operator.value.size.distrib.type = **constant**
operator.value.size.distrib.params.constant = **10**
event.generator.outoforder.percentage = **2**
event.generator.lateness.threshold = **3**
event.generator.key.popularity.distrib.type = **zipf**
event.generator.event.occurrence.distrib=**exponential**
event.generator.event.occurrence.distrib.lambda= **10**
watermark.frequency= **3**

Event Generator

watermark

Late events

Input replayer

Driver

event time clock

hIndex

Key partitioning

Windowing

vIndex

Workload Generator

event

Incremental window

Get → ε → Put → trigger → FGet → ε → Delete

Performance Evaluator

Trace replayer

RocksDB *Lethe* *FASTER* *BerkeleyDB* *Other KV store*

Gadget output

Ops: 1833698
Merge: 0
Put: 841135
Get: 916849
Delete: 75714

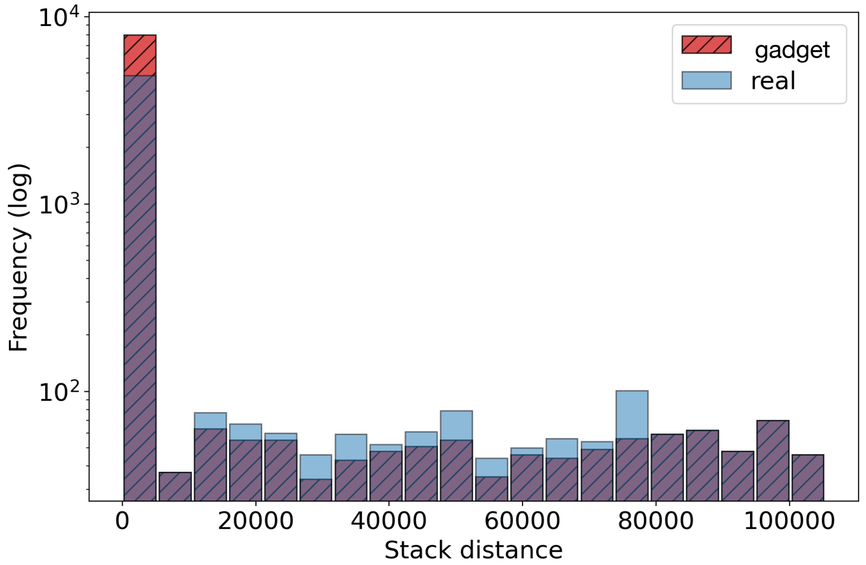
Ops/s:190771
P99.99: 1838 us
P99.9: 154 us
P99: 33 us

Max: 115511 us
Min: 0 us
AVG: 6.2165 us

Gadget can accurately simulate real state traces

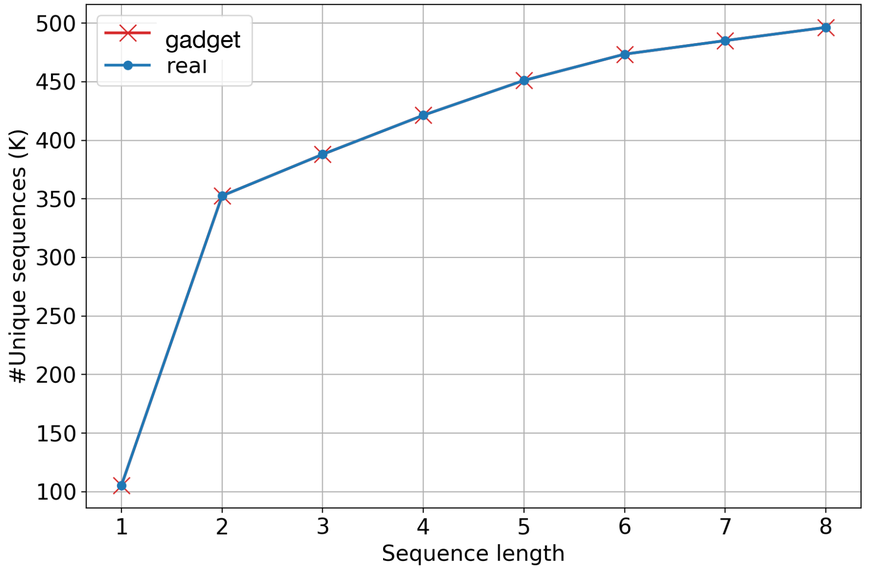
Temporal locality of synthetic and real traces

Sliding join



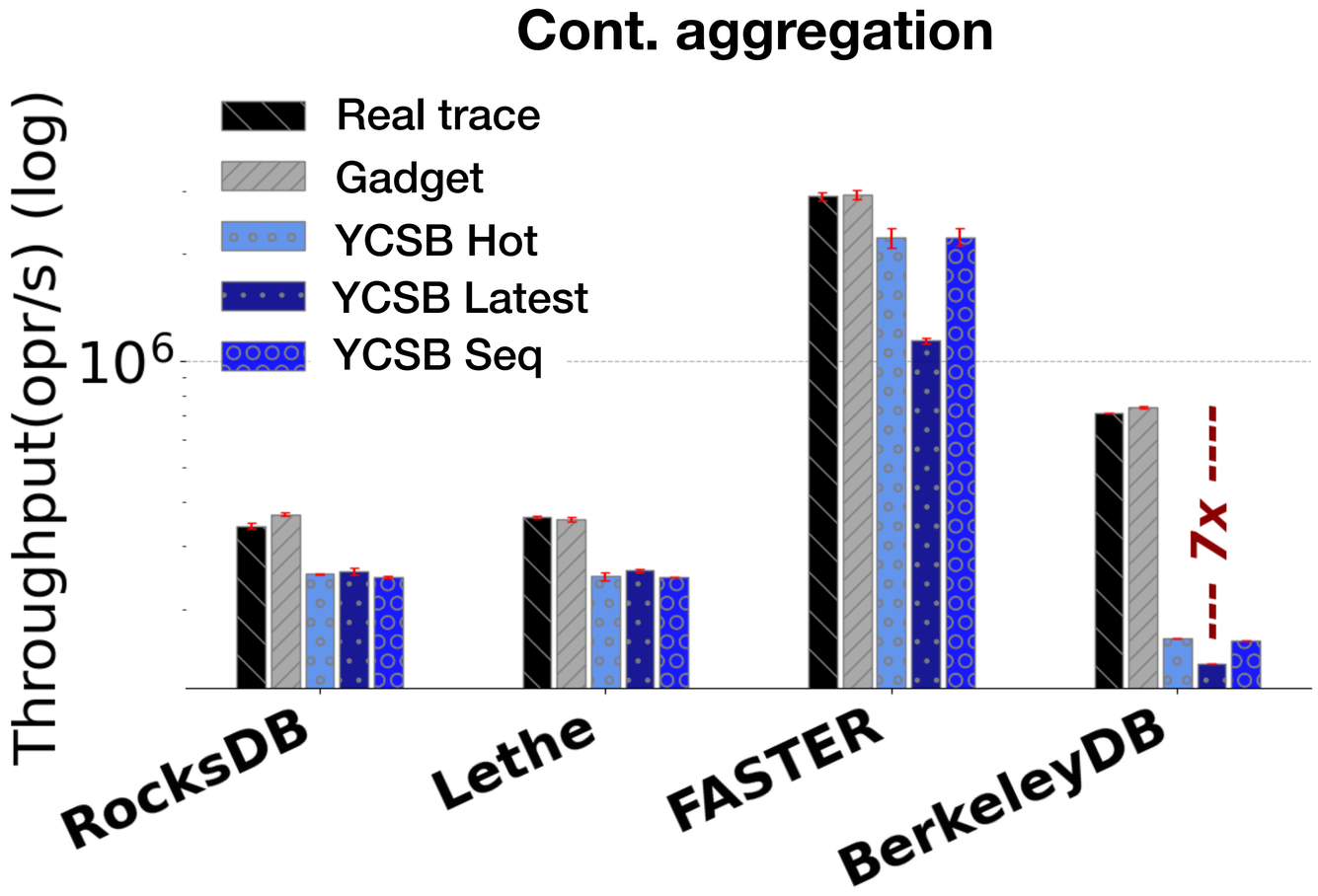
Stack distance: distribution of unique keys accessed between consecutive operations on the same key.

Spatial locality of synthetic and real traces



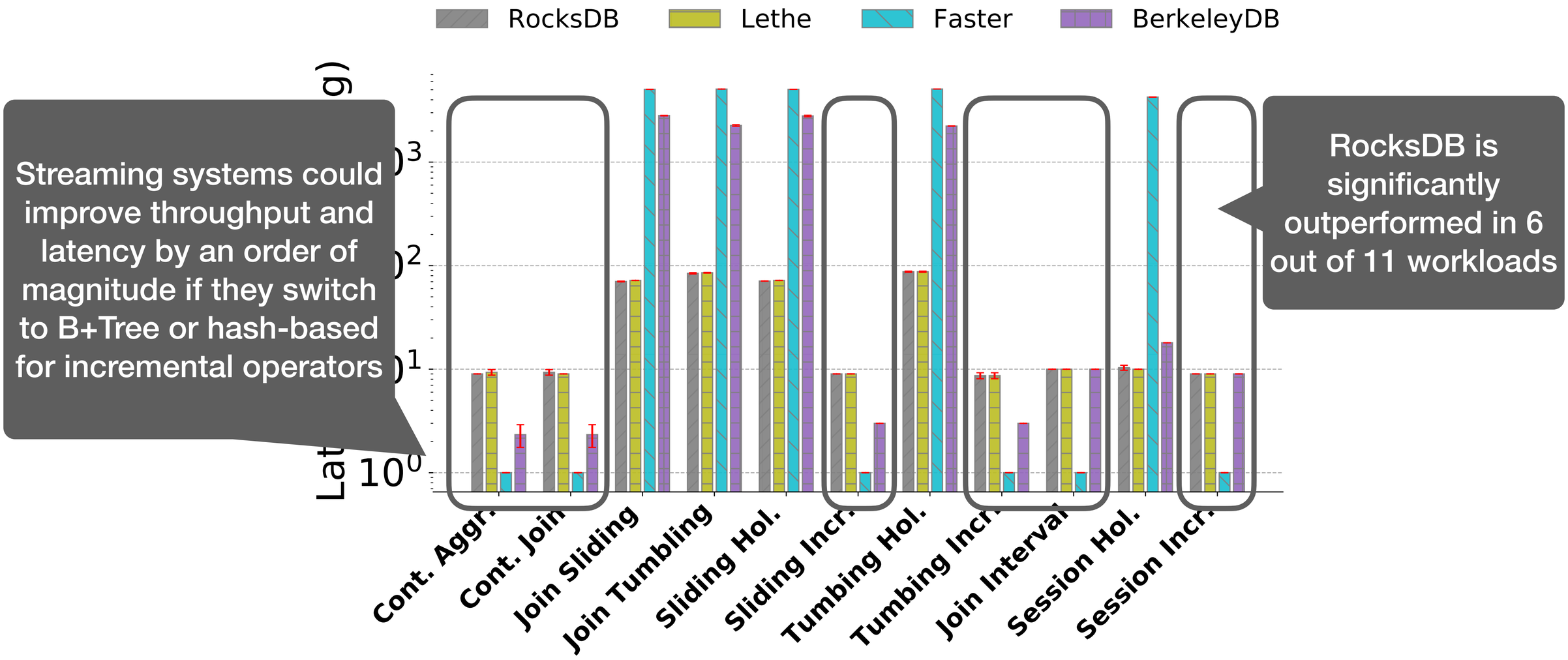
Unique sequences occurring in a state access stream with a maximum length of up to 8.

Gadget provides robust performance results



- YCSB **underestimates throughput** by up to 7x (BerkeleyDB).
- YCSB positions BerkeleyDB last: in reality it outperforms both RocksDB & Lethe.

RocksDB: a wide performance gap



Try it out!

https://github.com/CASP-Systems-BU/Gadget

Gadget enables **easier evaluation** of streaming state stores

- Automatic KV store configuration
- Novel store designs, e.g. hybrid or polystores
- Optimization of stateful operators: better windowing designs
- Alternative state management approaches, e.g. external KV stores

Gadget enables **easier tuning** of KV stores and streaming operators

- Trigger compactions according to delete frequency
- Automatic cache sizing, prefetching

A New Benchmark Harness for Systematic and Robust Evaluation of Streaming State Stores

Showan Asyabi, Yuanli Wang, John Liagouris, **Vasiliki Kalavri**, Azer Bestavros