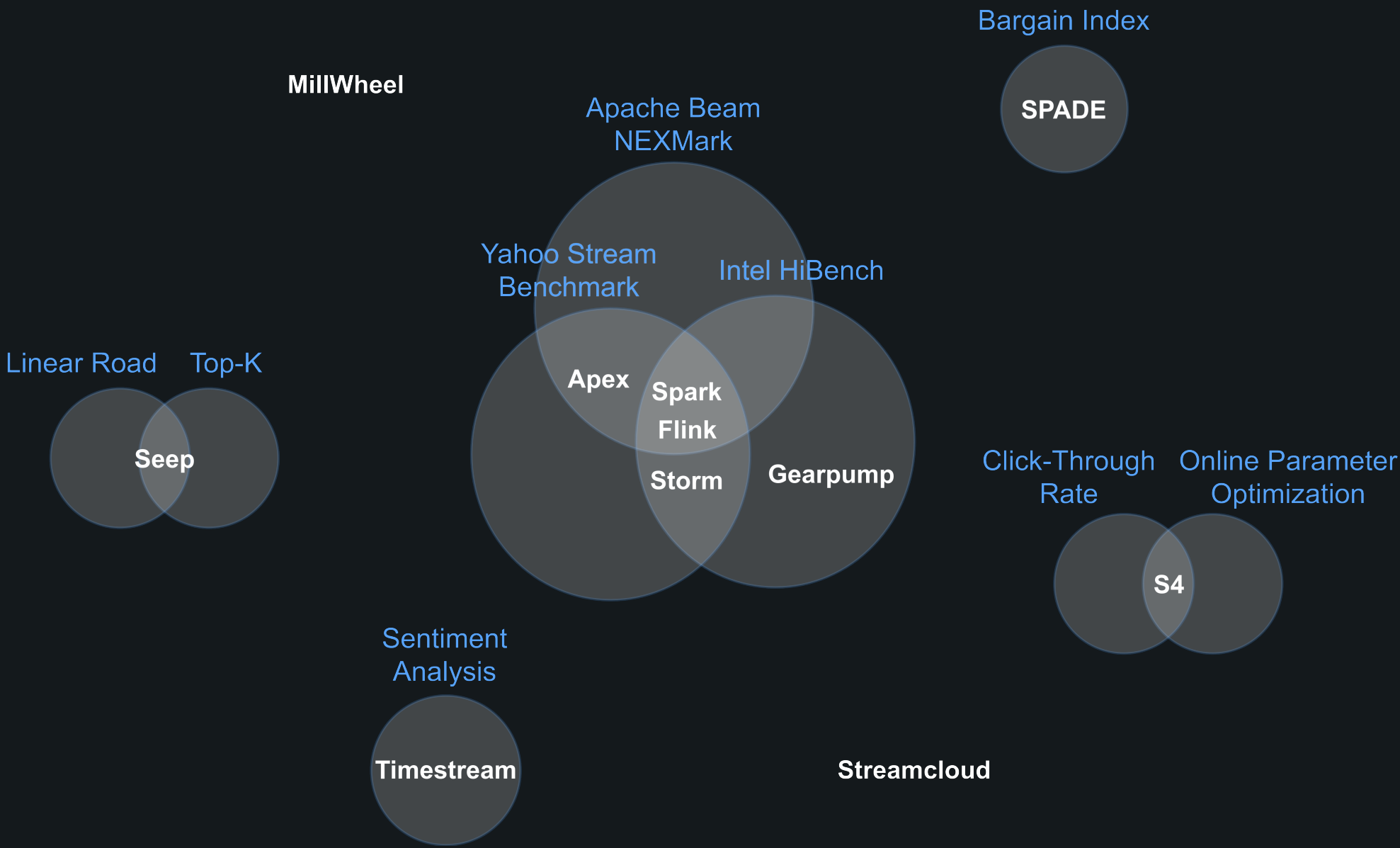
Implementation of a Benchmark Suite for Strymon

Nicolas Hafner

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

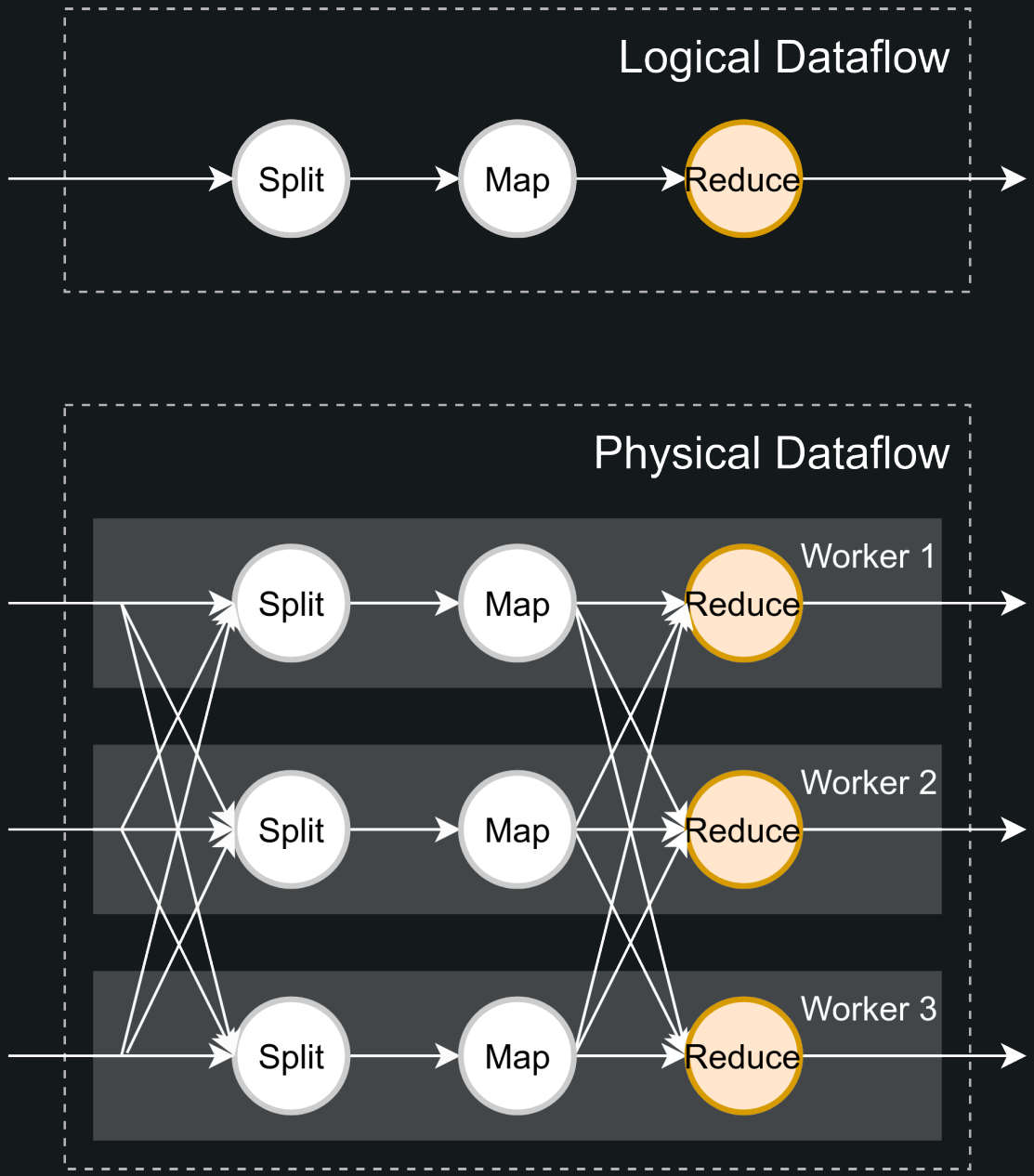
Systems@ETH Zürich


Bargain Index
MillWheel
SPADE
Apache Beam
NEXMark
Yahoo Stream
Benchmark
Intel HiBench
Linear Road
Top-K
Apex
Spark
Flink
Storm
Gearpump
Seep
Click-Through
Rate
Online Parameter
Optimization
S4
Sentiment
Analysis
Timestream
Streamcloud

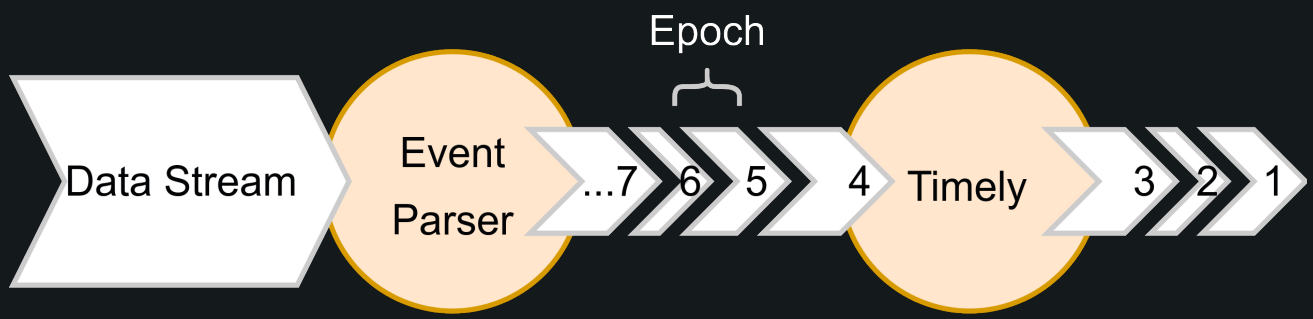
Current Publications

- Investigated current practises in published papers
- Almost no paper used a standardised benchmark
- Code and data often not published
- Often very simple benchmarks like Word Count:

Timely


Logical Dataflow
Split
Map
Reduce
Physical Dataflow
Worker 1
Split
Map
Reduce
Worker 2
Split
Map
Reduce
Worker 3
Split
Map
Reduce

Timely



Benchmarks

- We implemented three benchmarks:

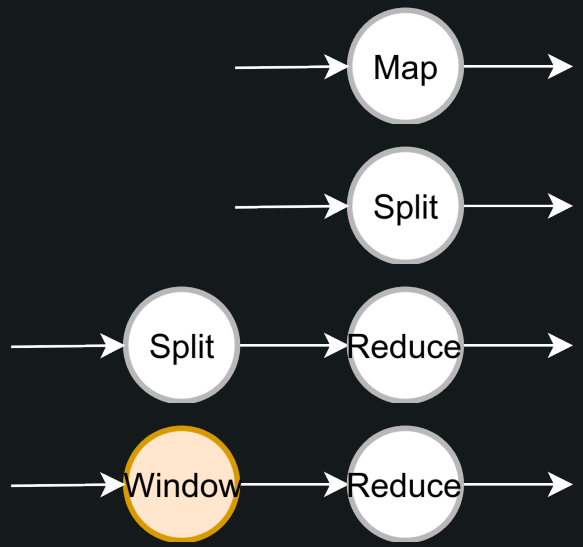
1. Intel's HiBench
2. Yahoo's Streaming Benchmark
3. Apache Beam's NEXMark

- Comparable against many other systems

Intel's HiBench[1]

- Big Data micro-benchmark
- Only four data flows:

1. Identity
2. Repartition
3. Word Count
4. Window Reduce



[1] https://github.com/intel-hadoop/HiBench

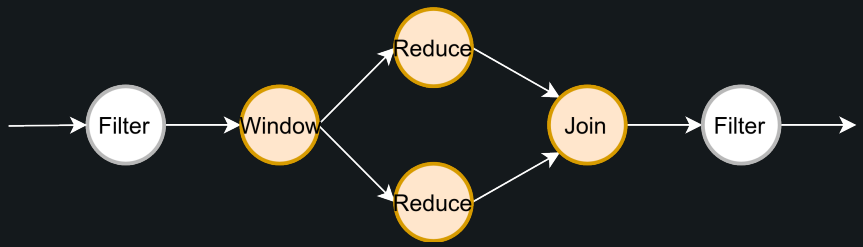
Yahoo Stream Benchmark[2]

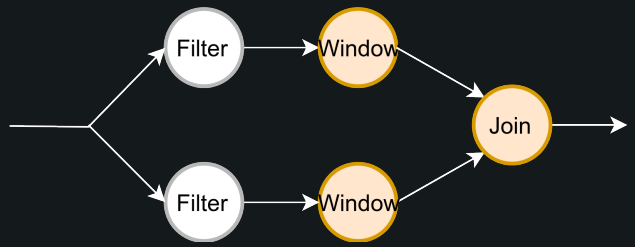
- Count ad views for ad campaigns
- Only one, relatively simple data flow:

[2] Sanket Chintapalli et al. "Benchmarking streaming computation engines: Storm, Flink and Spark streaming". In: Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International. IEEE. 2016, pp. 1789–1792.

Beam's NEXMark[3]

- Implements an "auctioning system"
- 13 data flows in total
- Uses filter, map, reduce, join, window, session, partition
- Dataflows for Query 5 and 8:





[3] Based on original paper: Pete Tucker et al. NEXMark–A Benchmark for Queries over Data Streams (DRAFT). Tech. rep. Technical report, OGI School of Science & Engineering at OHSU, Septembers, 2008.

Testing Framework

- Implemented a general framework for benchmarks
- Generic components for input/output handling
- New, reusable operators for join, window, reduce, filtermap, session, partition
- Implemented HiBench, YSB, NEXmark using this framework

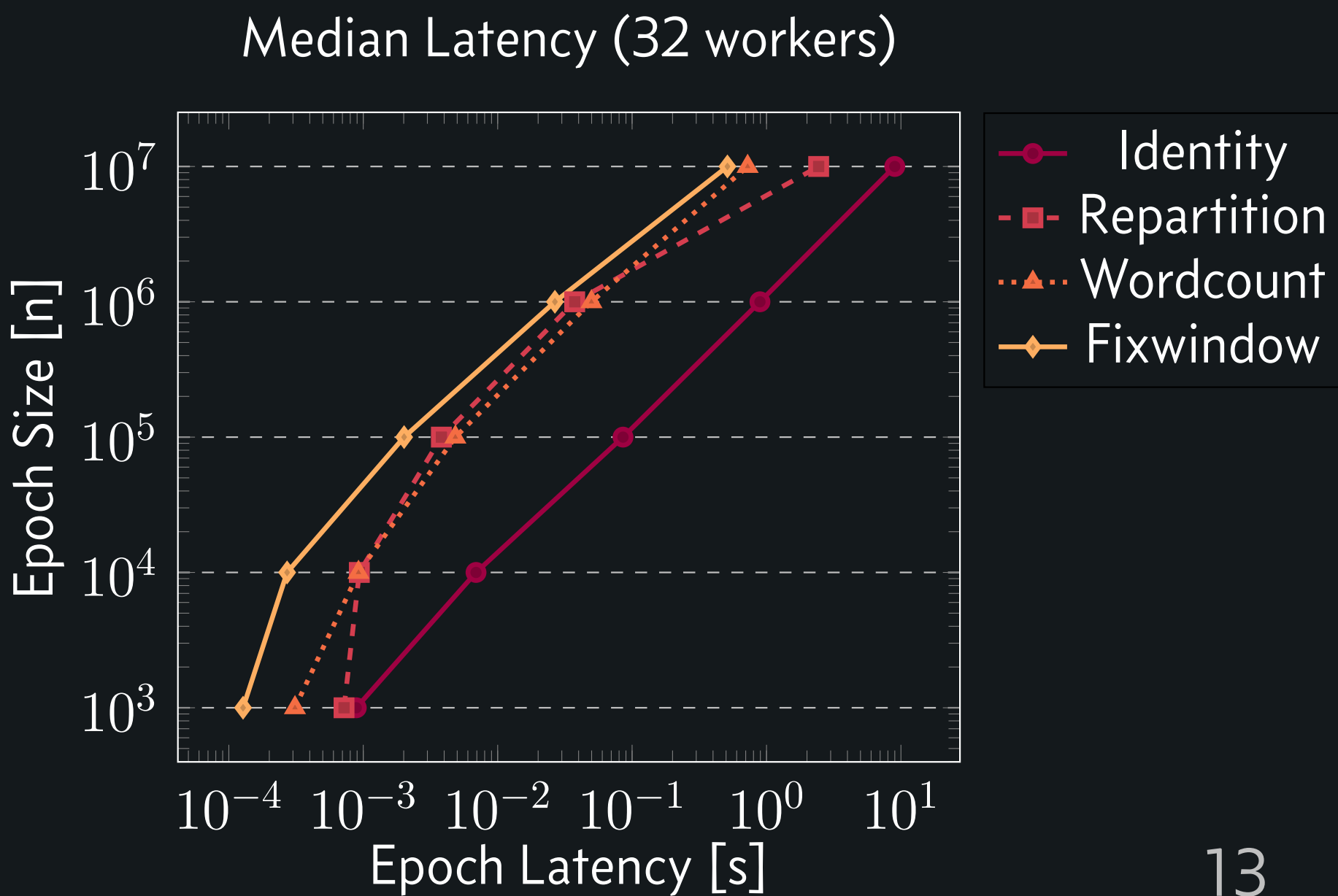
Evaluation System

- Run on sgs-r815-03 (AMD, 64 cores, 2.4GHz)
- Data generated directly in memory
- Generation re-implemented in Rust
- No foreign systems like Kafka used

Evaluation Setup

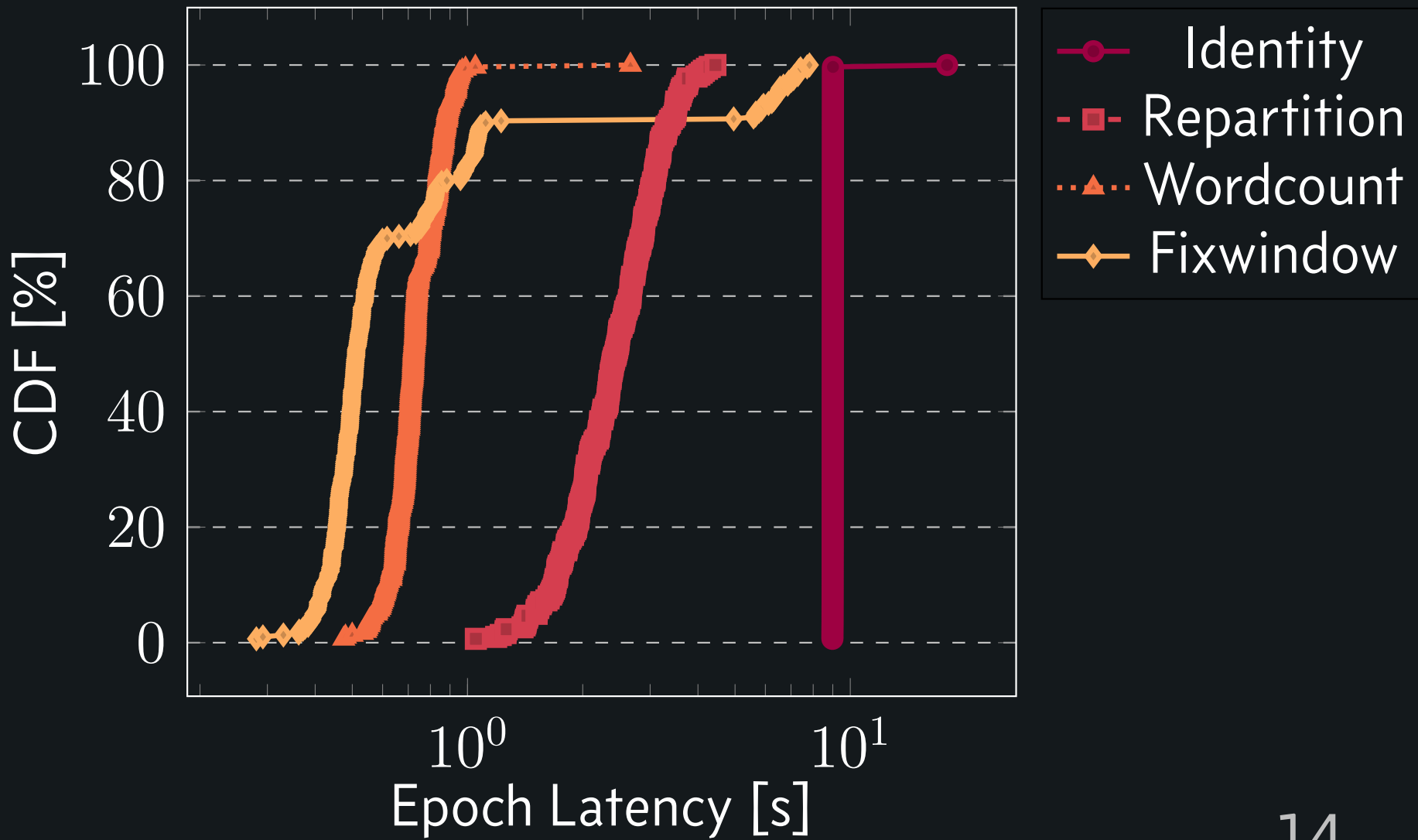
- Measured closed-loop per-epoch latency
- One epoch encompasses a logical second of data
- Workload varied between 1K-10Me/s, 1-32 workers, 10-120s windows

HiBench Latency / Batch

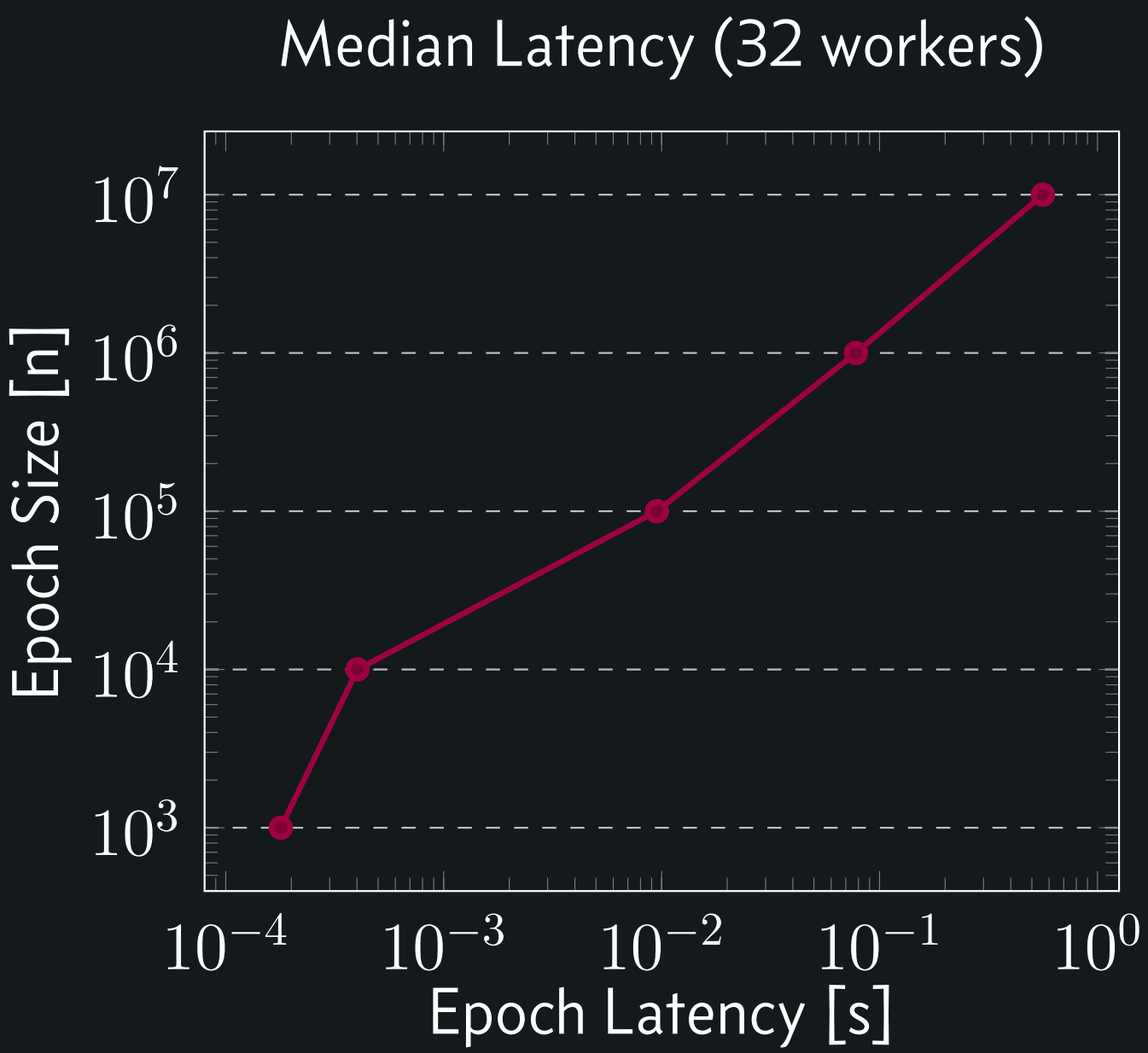


HiBench CDF

CDF (32 workers, 10′000′000 events/epoch)

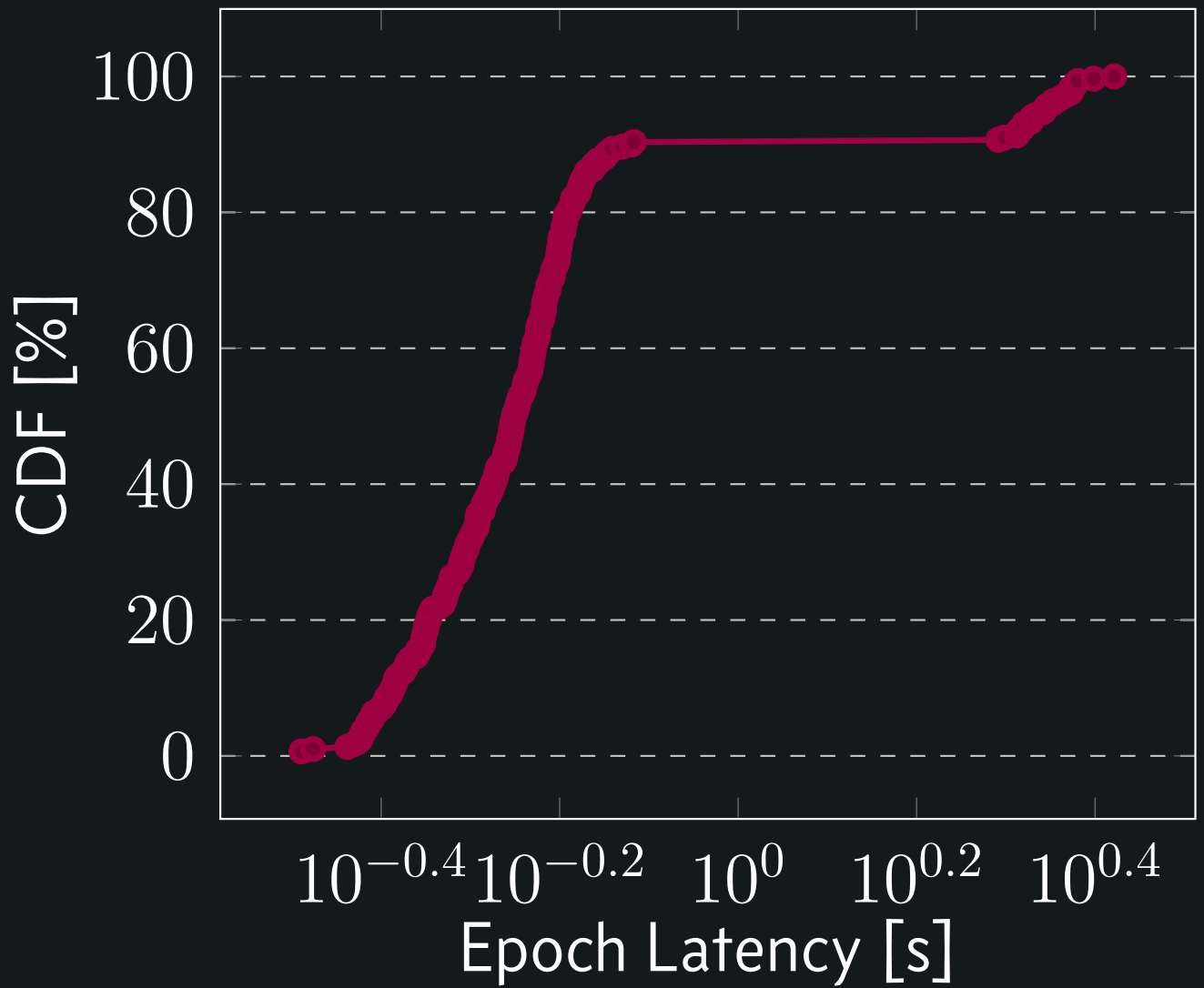


YSB Latency / Batch

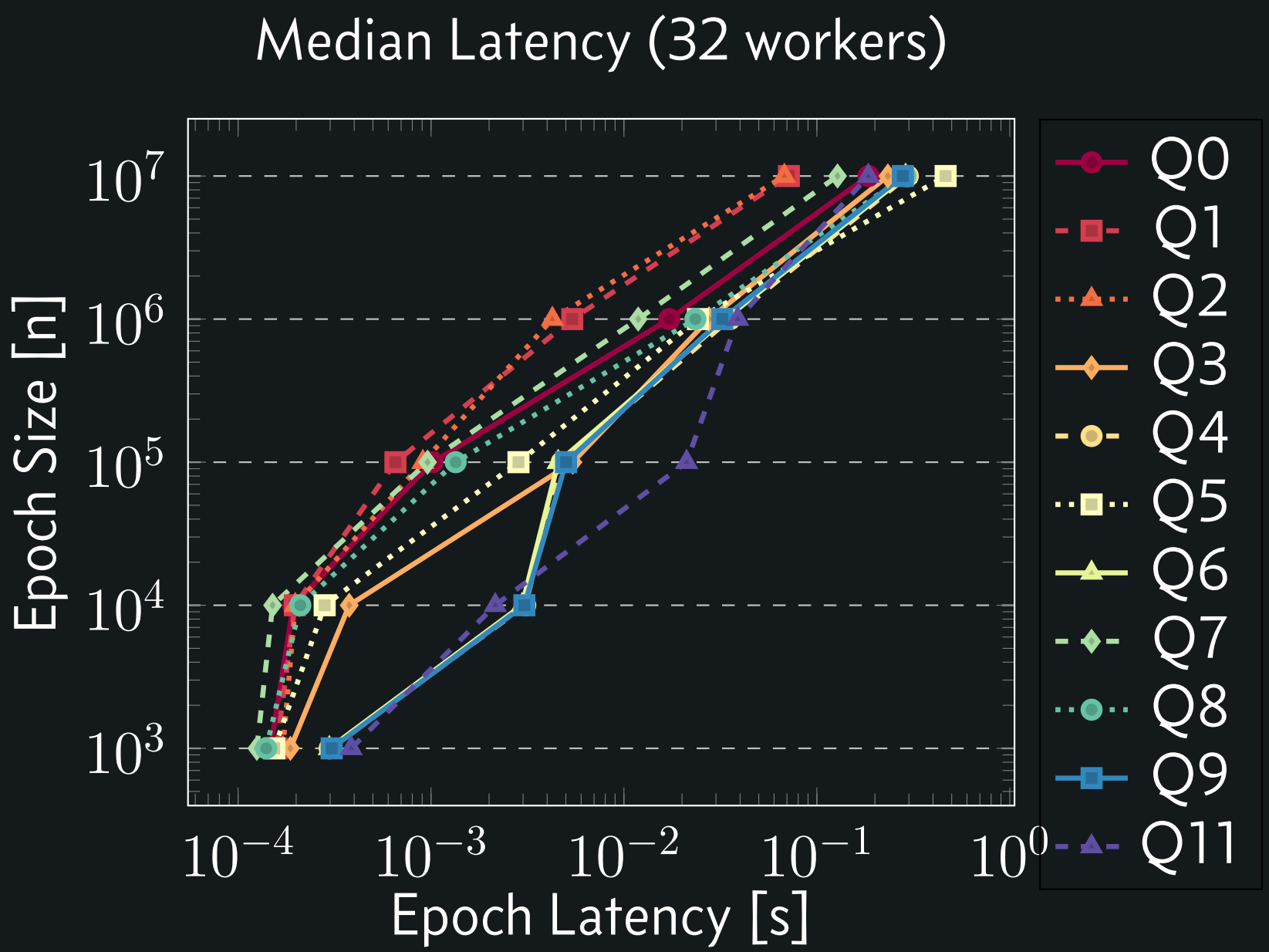


YSB CDF

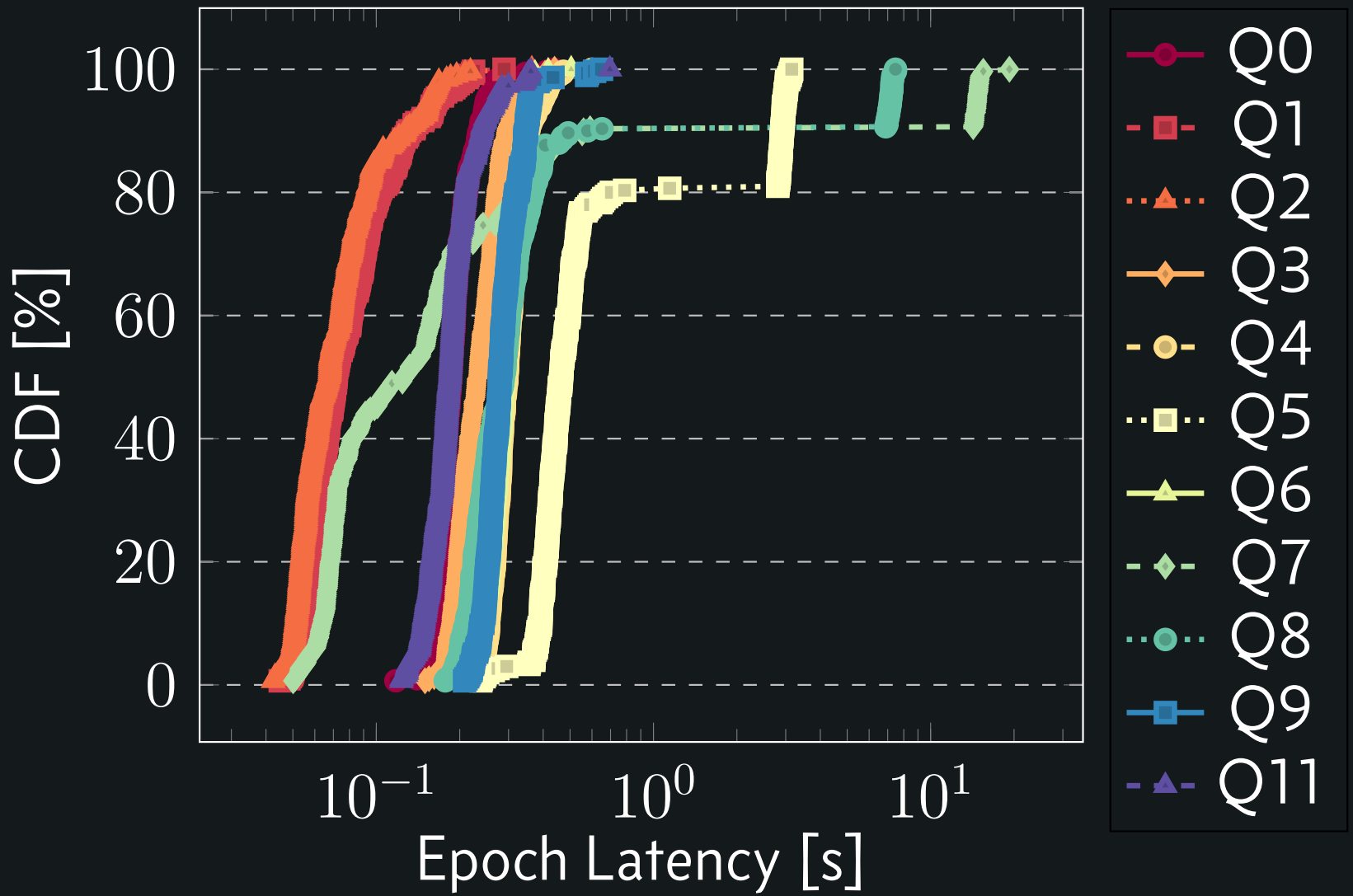
CDF (32 workers, 10′000′000 events/epoch)



NEXMark Latency / Batch



NEXMark CDF


CDF (32 workers, 10′000′000 events/epoch)
CDF [%]
100
80
60
40
20
0
10^{-1}
10^{0}
10^{1}
Epoch Latency [s]
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
Q9
Q11

Benchmark Evaluation

- Benchmarks are underspecified and undocumented

Benchmark Evaluation

- Benchmarks are underspecified and undocumented
- No result verification

Benchmark Evaluation

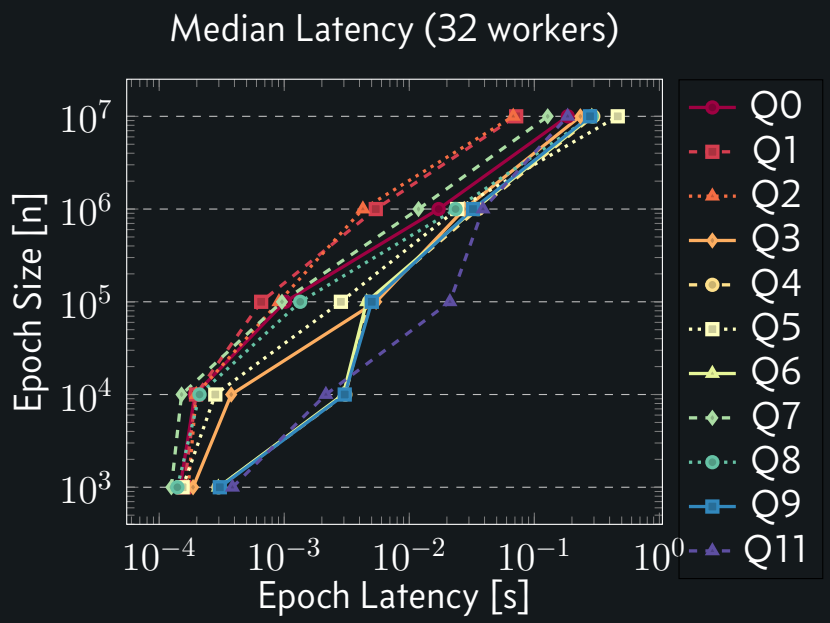
- Benchmarks are underspecified and undocumented
- No result verification
- External systems compound complexity

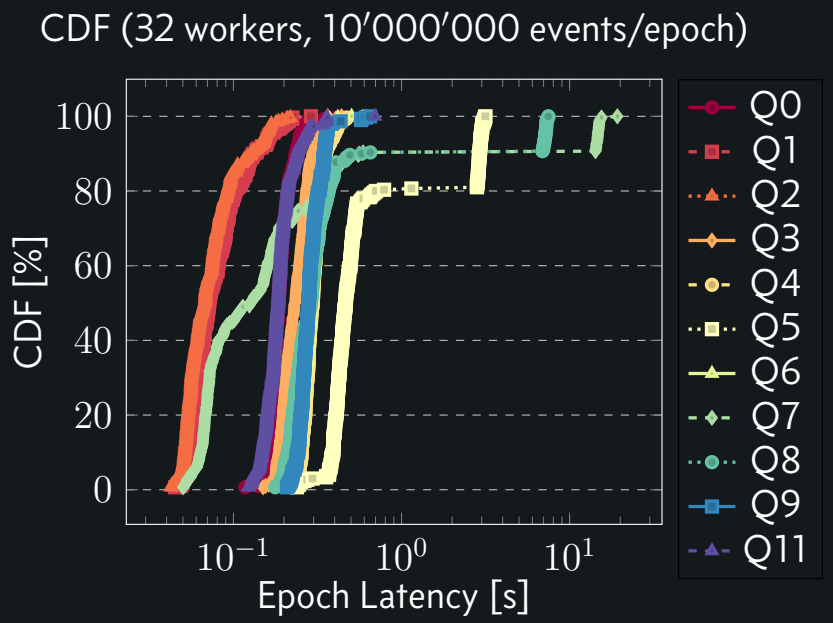
Benchmark Evaluation

- Benchmarks are underspecified and undocumented
- No result verification
- External systems compound complexity
- No tests for load balancing, fault-tolerance, etc.

Benchmark Suggestions

- Abstract model definitions for data flows
- Correctness verification tools
- Deterministically generated workloads
- Various short and long data flows
- Tests for both latency, *and* fault-tolerance, etc.

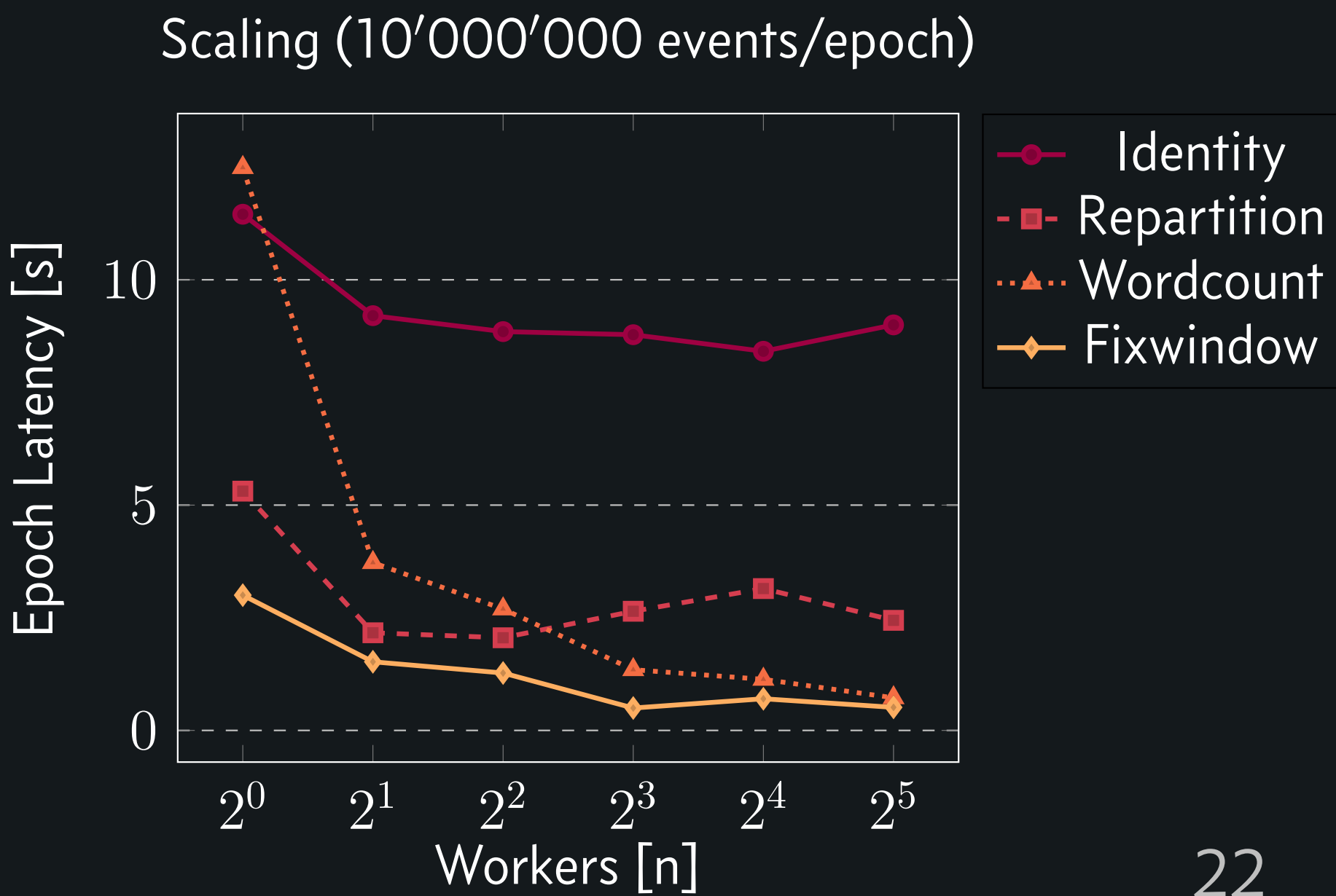




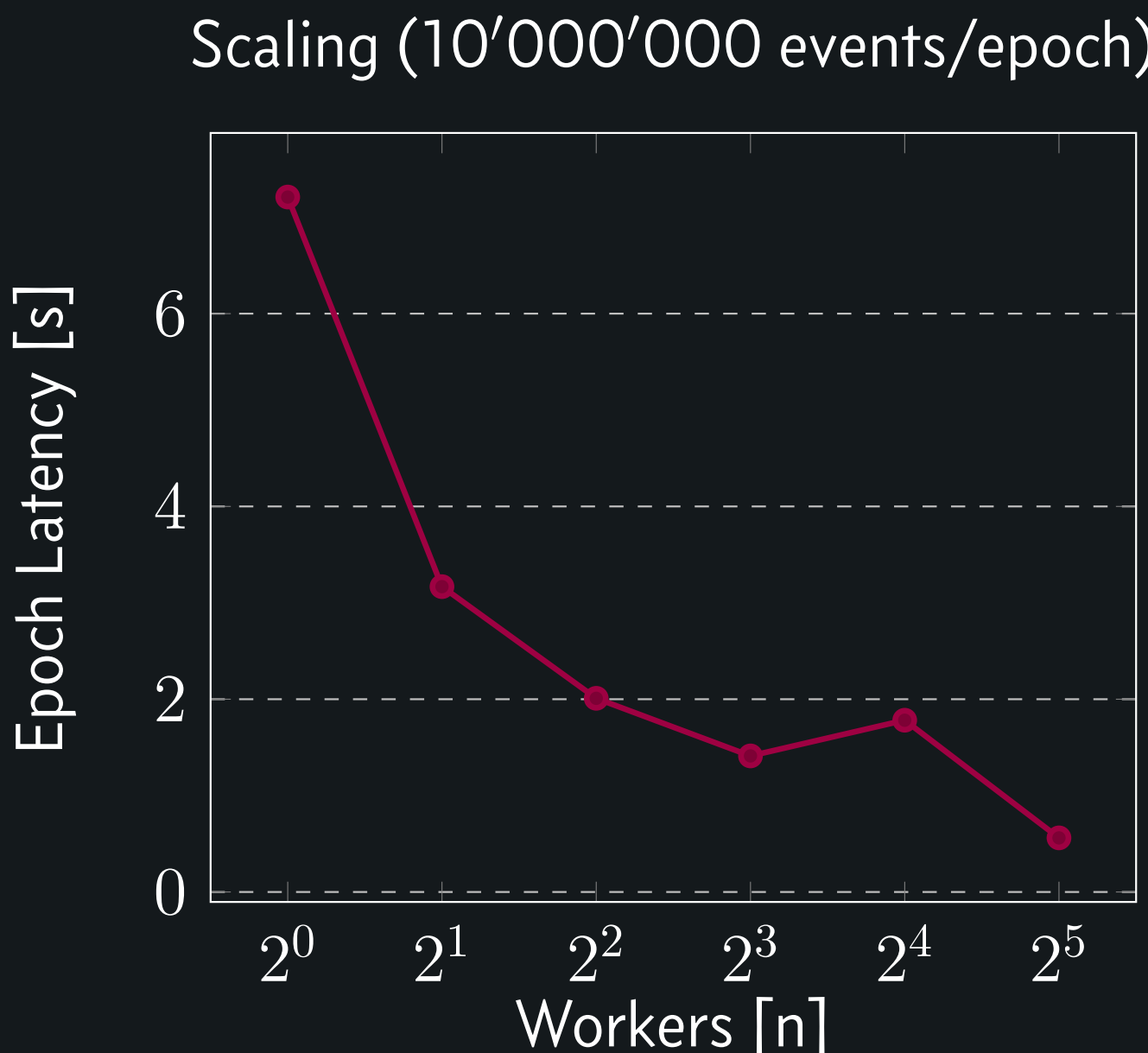
http://strymon.systems.ethz.ch/

https://github.com/Shinmera/bsc-thesis

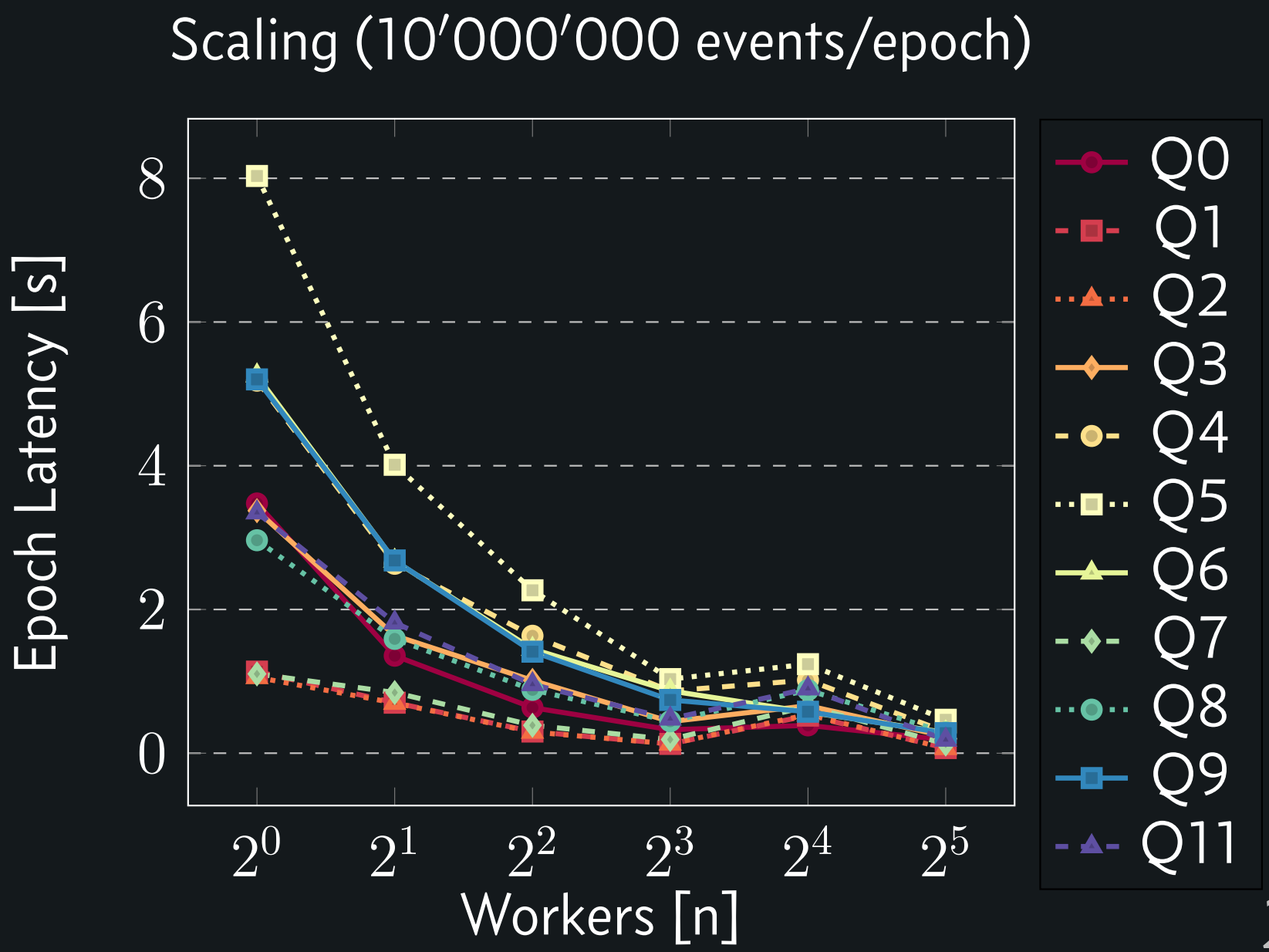
HiBench Worker Scaling



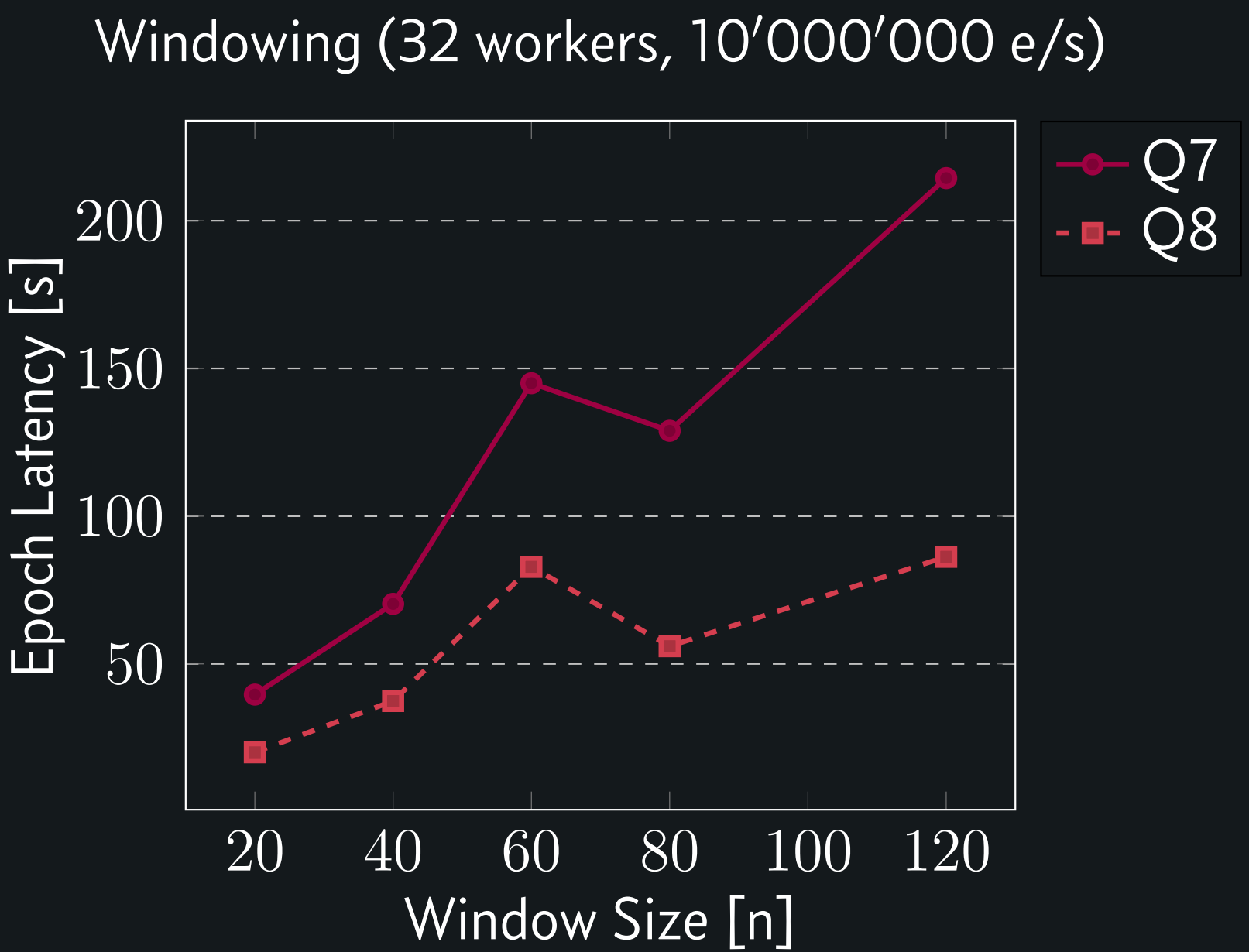
YSB Worker Scaling



NEXMark Worker Scaling



NEXMark Window Scaling



NEXMark Slide Scaling

Window Slides Q5 (32 workers, 10'000'000 e/s)