CompressStreamDB:

Fine-Grained Adaptive Stream Processing without Decompression

Renmin University of China

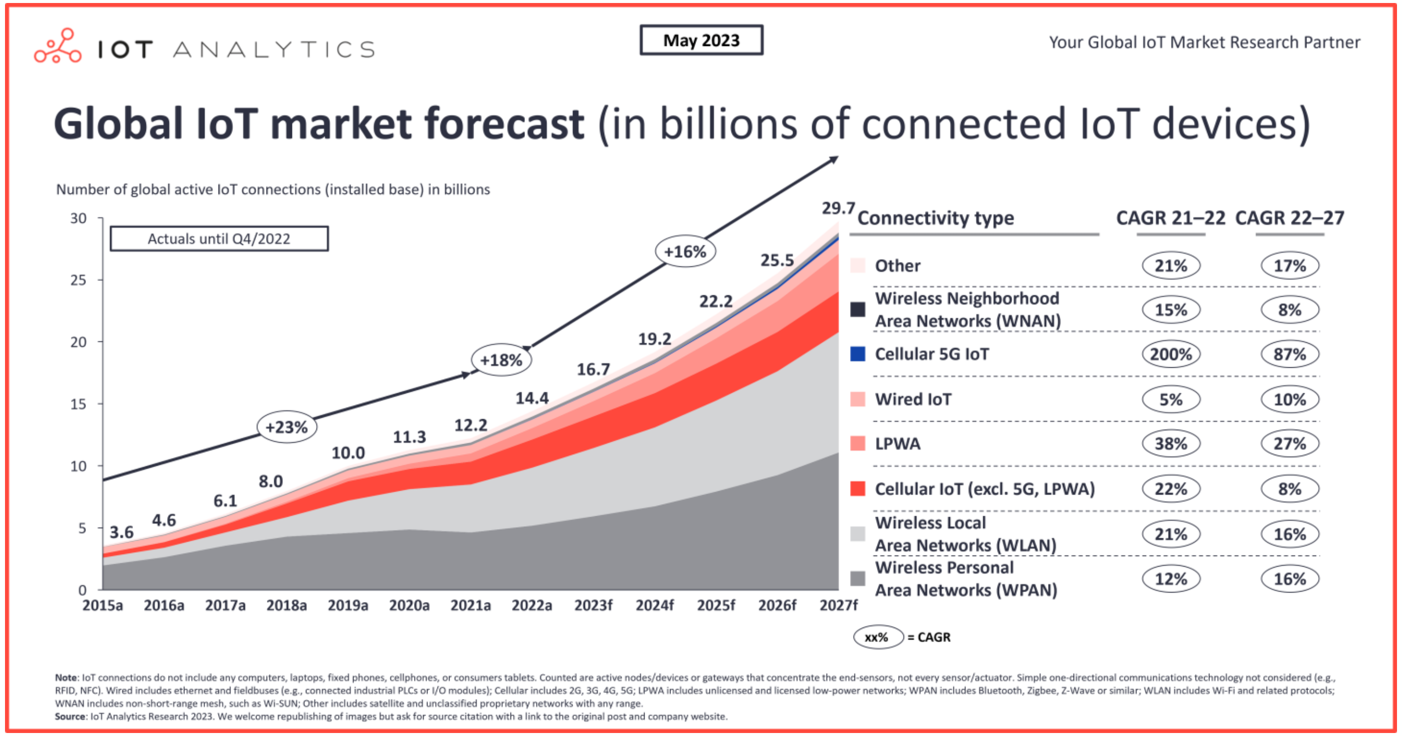
Singapore University of Technology

2023 IEEE 39th International Conference on Data Engineering (ICDE)

PURDUE UNIVERSITY

The Growth Of IoT

- 12.3 billion IoT endpoints (2021)
- Data
  - Sensor data
  - Financial transactions
  - Etc.



“State of IoT 2021,” https://iot-analytics.com/number-connected-iot-devices/, 2021.

Time Breakdown for Uncompressed Streams

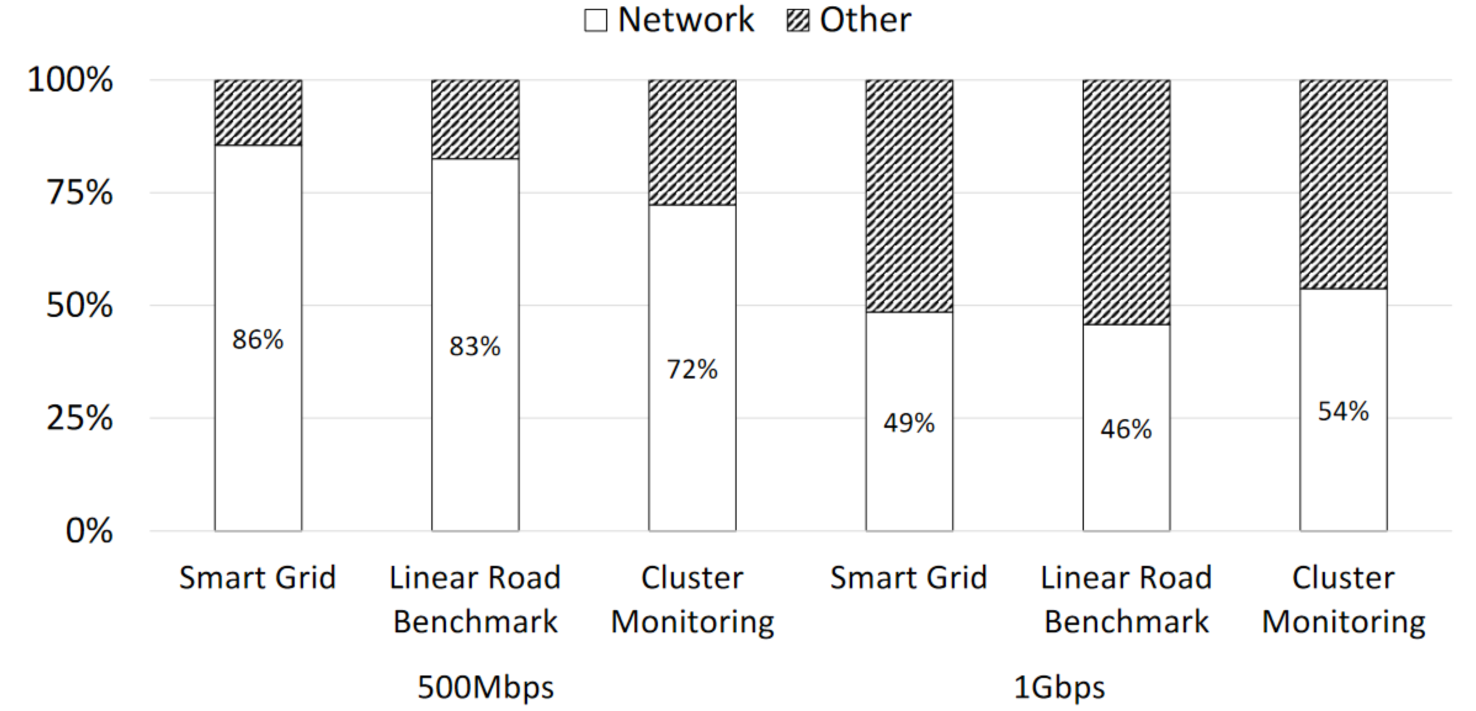


Fig. 3. Total time breakdown.

Linear Road System

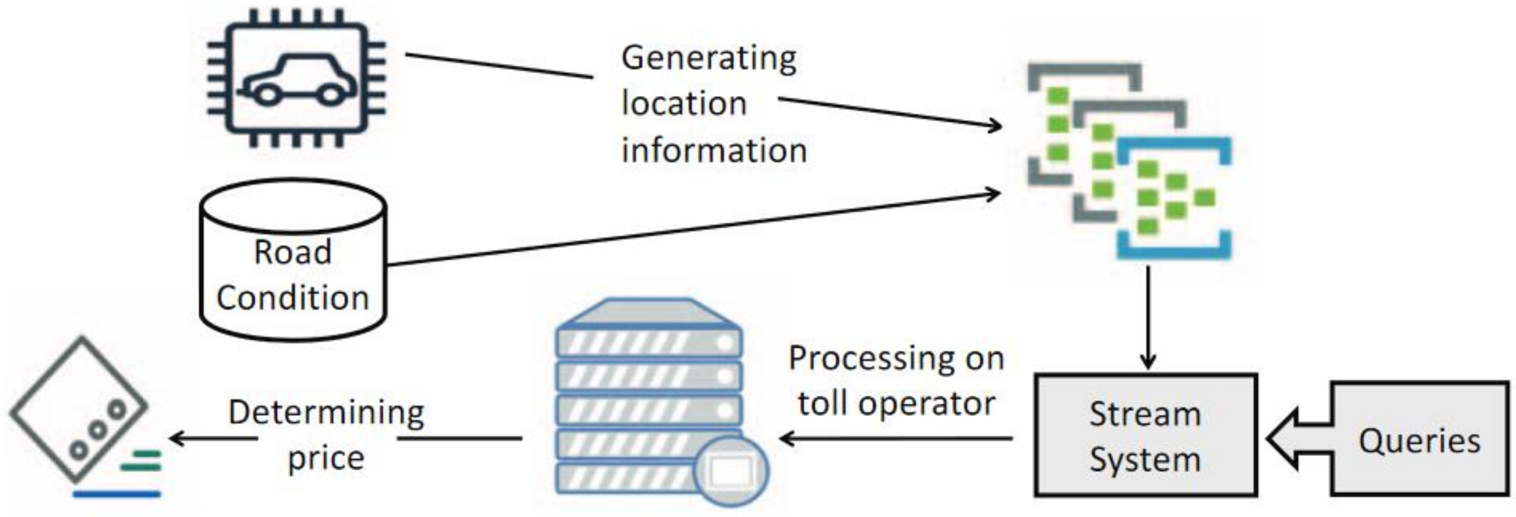


Fig. 1. Use case of linear road system.

PURDUE UNIVERSITY

Stream Processing → Streaming SQL

Stream Processing

- "Real-time processing of continuous streams of data, events, and messages."
- Applied to:
  - Dataflow systems
  - Reactive systems
  - Real-time systems

Streaming SQL

- Extension of stream processing
- Query data streams continuously instead of all at once

PURDUE UNIVERSITY

Compression Algorithms

Choosing the right algorithms

- Lossless vs ~~Lossy~~
  - Accuracy required
- Lightweight vs ~~heavyweight~~
  - Need minimal (de)compression overhead
- Eager and lazy compression algorithms considered
  - Eager: compress when a tuple arrives
  - Lazy: compress after waiting for an entire batch

Eager and Lazy Compression Methods in Lightweight Compression

	Compression Method	Description
Eager	Elias Gamma Encoding	Encode each value with unary and binary bits.
E	Elias Delta Encoding	Encode each value with unary and binary bits.
E	Null Suppression with Fixed Length	Delete leading zeros of each value with fixed bits.
E	Null Suppression with Variable Length	Delete leading zeros of each value with variable bits.
Lazy	Base-Delta Encoding	Encode values as their delta values from base value
L	Run Length Encoding	Encode values with their run lengths.
L	Dictionary	Maintain a dictionary of the distinct values.
L	Bitmap	Encode each distinct value as a bit-string.

Table I

CompressStreamDB Framework

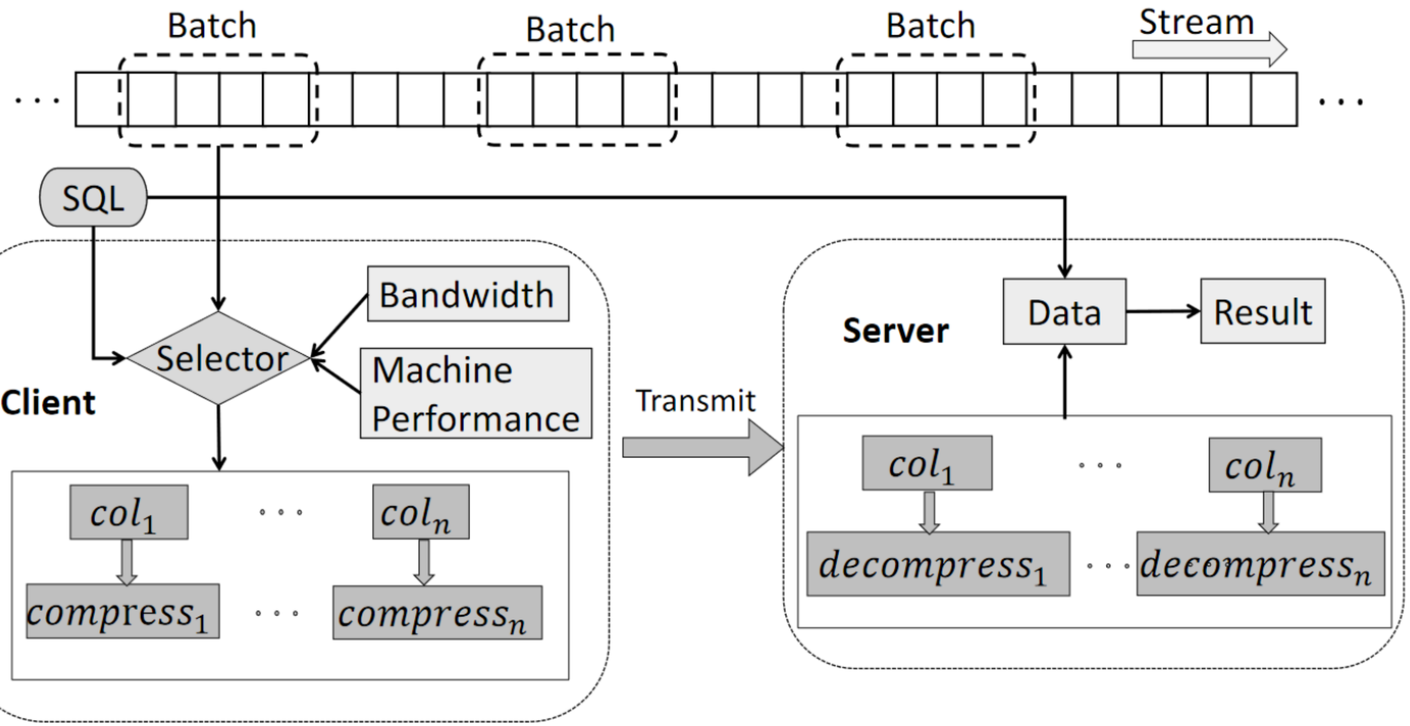


Fig. 4. CompressStreamDB framework.

Compressed Stream Processing

- Compression
  - 8 different algorithms
    - 4 fixed length
    - 4 variable length
- Adaptive processing for dynamic workload
  - Use SQL for processing
  - Algorithms are reselected after a preset number of batches
- Query without decompression
  1. Compressed data is similar to the original data
  2. Compressed stream data is aligned
  3. Compression does not affect the order of the stream

Example:

Pre-Compression	Post-Compression
col1 8 bytes	col1’ 2 bytes
col2 4 bytes	col2’ 1 bytes
col3 4 bytes	col3’ 1 bytes

SELECT col1, AVG(col2) FROM data GROUP BY col3;

MAPPED TO:

SELECT col1’, AVG(col2’) FROM data GROUP BY col3’;

System Cost Model

4 Stages

- Compression
- Transmission
- Decompression*
- Query Processing

TABLE II
SYMBOLS AND MEANINGS.

Symbol	Description
α	The compression algorithm is lazy or eager.
β	Whether the compression needs decompression.
r	The compression ratio in transmission step.
r'	The compression ratio in query step.
τ	The compression algorithm.
$Size_T$	The number of bytes per tuple.
$Size_B$	The number of tuples per batch.
$N_{client} \& N_{server}$	The machine performance.
$T_{memory}^{com,\tau} \& T_{memory}^{decom,\tau}$	The number of instructions for memory accesses.
$T_{operation}^{com,\tau} \& T_{operation}^{decom,\tau}$	The number of instructions for computation.

* Decompression is not always necessary

PURDUE UNIVERSITY

System Cost Model

Stream Processing

System Cost:

$$t = t_{compress} + t_{trans} + t_{decom} + t_{query} \tag{1}$$

Compression Time:

$$t_{compress} = \alpha \cdot t_{wait} + \frac{T^{com,\tau}_{memory} + T^{com,\tau}_{operation}}{N_{client}} \tag{2}$$

Transmission Time:

$$t_{trans} = \frac{Size_T \cdot Size_B}{r} \cdot latency \tag{4}$$

Decompression Time:

$$t_{decompress} = \beta \cdot \frac{T^{decom,\tau}_{memory} + T^{decom,\tau}_{operation}}{N_{server}} \tag{6}$$

Query Time:

$$t_{query} = t^{query}_{operation} + \frac{t^{query}_{memory}}{r'} \tag{8}$$

TABLE II
SYMBOLS AND MEANINGS.

Symbol	Description
α	The compression algorithm is lazy or eager.
β	Whether the compression needs decompression.
r	The compression ratio in transmission step.
r'	The compression ratio in query step.
τ	The compression algorithm.
$Size_T$	The number of bytes per tuple.
$Size_B$	The number of tuples per batch.
$N_{client}\&N_{server}$	The machine performance.
$T^{com,\tau}_{memory}\&T^{decom,\tau}_{memory}$	The number of instructions for memory accesses.
$T^{com,\tau}_{operation}\&T^{decom,\tau}_{operation}$	The number of instructions for computation.

$$\alpha = \begin{cases} 1, & \text{if the compression algorithm } \tau \text{ is lazy;} \\ 0, & \text{if the compression algorithm } \tau \text{ is eager.} \end{cases} \tag{3}$$

$$\beta = \begin{cases} 1, & \text{if the compression algorithm } \tau \text{ needs decompression;} \\ 0, & \text{otherwise .} \end{cases} \tag{7}$$

$$r' = \begin{cases} 1, & \text{if the compression algorithm needs decompression;} \\ r, & \text{otherwise.} \end{cases} \tag{9}$$

PURDUE UNIVERSITY

Implementation

Client - Server

- Client
  - Compression algorithms
  - Adaptive selector
- Server
  - SQL operators
  - Processing compressed streams
  - Profiler to collect performance data
    - (de)compression
    - transmission time

Evaluation

- Baseline: CompressStreamDB without compression
- Platform: Client & Server
  - Intel Xeon Platinum 8269CY
  - 2.5 GHz CPU
  - 16GB memory
  - Ubuntu 20.04.3 LTS
  - Java 8
  - Network from 0 to 1Gbps between client & server
- Datasets:
  - Energy consumption measurement in smart grids
  - Compute cluster monitoring
  - Linear road benchmark

Evaluation

Benchmarks

TABLE III
THE QUERIES USED IN EVALUATION.

Query	Detail
Q1	`select timestamp, avg (value) as globalAvgLoad from SmartGridStr [range 1024 slide 1]`
Q2	`select timestamp, plug, household, house, avg(value) as localAvgLoad from SmartGridStr [range 1024 slide 1] group by plug, household, house`
Q3	`(select timestamp, vehicle, speed, highway, lane, direction, (position/5280) as segment from PosSpeedStr [range unbounded]) as SegSpeedStr -- select distinct L.timestamp, L.vehicle, L.speed, L.highway, L.lane, L.direction, L.segment from SegSpeedStr [range 30 slide 1] as A, SegSpeedStr [partition by vehicle rows 1] as L where A.vehicle == L.vehicle`
Q4	`select timestamp, avg(speed), highway, lane, direction from PosSpeedStr [range 1024 slide 1] group by highway,lane,direction`
Q5	`select timestamp, category, sum(cpu) as totalCPU from TaskEvents [range 512 slide 1] group by category`
Q6	`select timestamp, eventType, userId, max(disk) as maxDisk from TaskEvents [range 512 slide 1] group by eventType, userId`

PURDUE UNIVERSITY

Performance Comparison

Throughput

- On average, CompressStreamDB improves throughput by 3.24× across all datasets and queries
- Smart Grid dataset: 4.80× faster than the baseline, with DICT encoding providing a 3.00× improvement
- Linear road benchmark dataset: 2.38× throughput improvement compared to the baseline, outperforming NS by 4.4%
- Google Cluster Monitoring dataset: 2.55× throughput improvement, surpassing Base-Delta by 8.1%

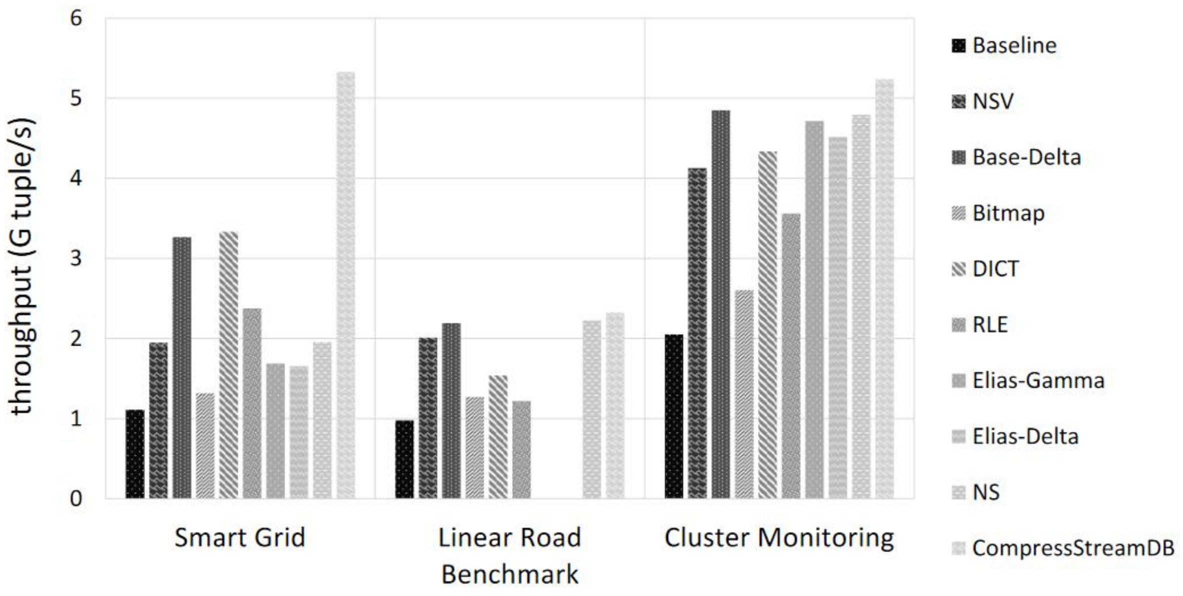


Fig. 5. Throughput of different compression methods.

PURDUE UNIVERSITY

Performance Comparison

Latency

- On average, CompressStreamDB achieves a significant 66.0% reduction in latency across all datasets
- Smart Grid dataset: CompressStreamDB demonstrates a 79.2% lower latency compared to the uncompressed system and a 37.5% improvement over DICT encoding
- Linear road benchmark dataset: CompressStreamDB shows a 58.0% lower latency compared to the baseline, with a 4.2% improvement over NS
- Google Cluster Monitoring dataset: CompressStreamDB achieves a 60.8% reduction in latency compared to the baseline, outperforming Base-Delta by 7.4%.

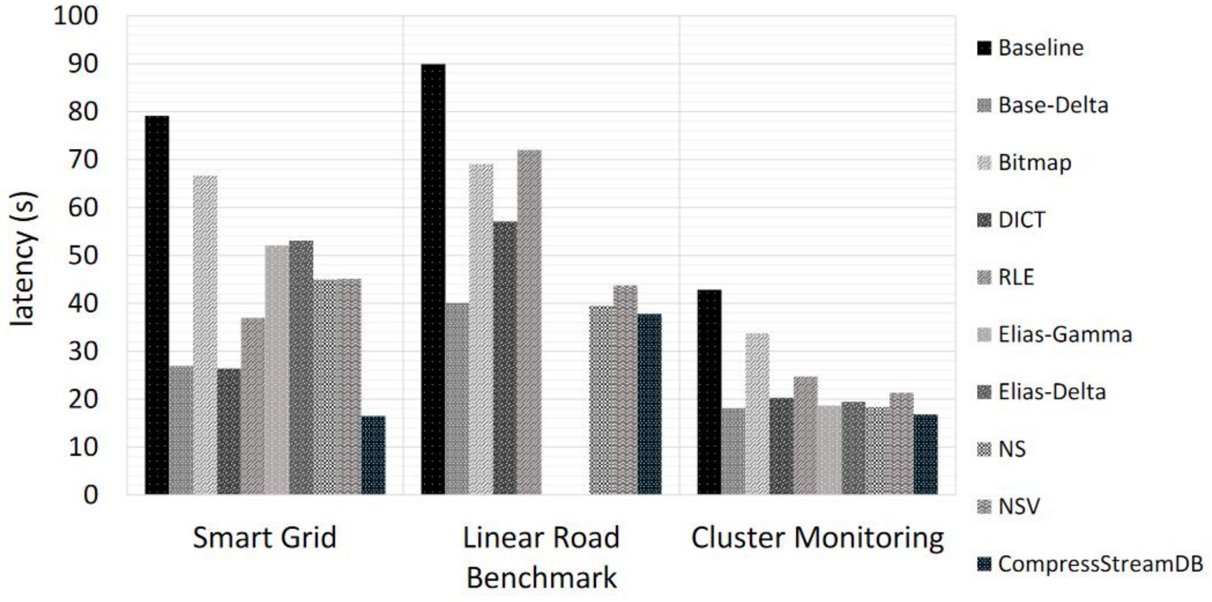


Fig. 6. Latency of different compression methods.

PURDUE UNIVERSITY

Performance Comparison

Dynamic Workload

- 100Mbps speed has the highest performance improvement against the baseline
- Optimal Static Method: 3.97x speedup
- CompressStreamDB: 9.68x speedup

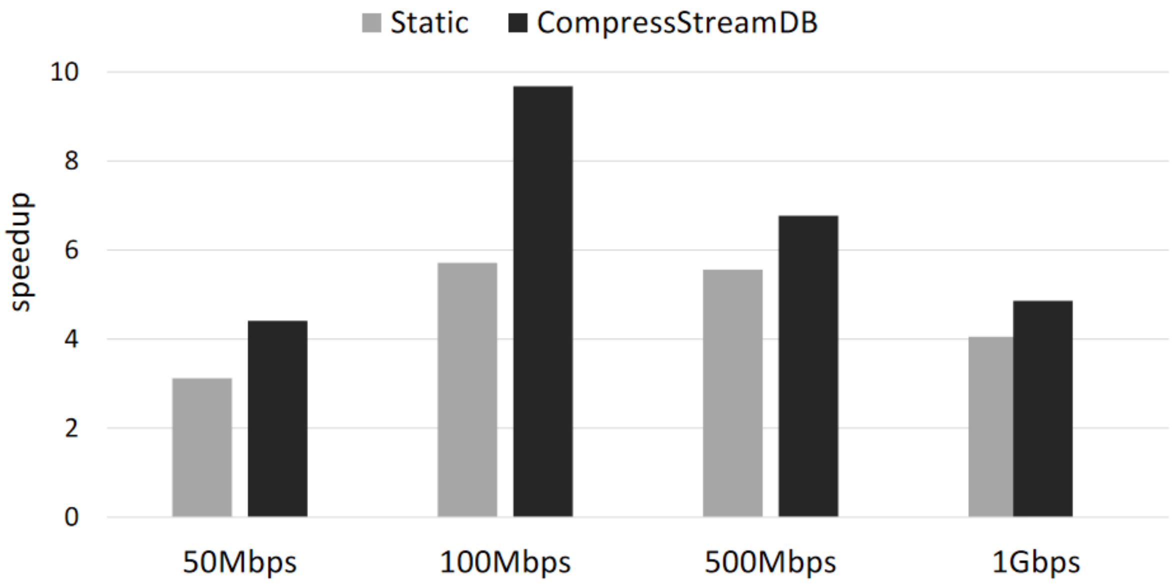


Fig. 7. Speedup with dynamic workload.

PURDUE UNIVERSITY

Model Accuracy

- On average, the CompressStreamDB cost model is 88.2% accurate

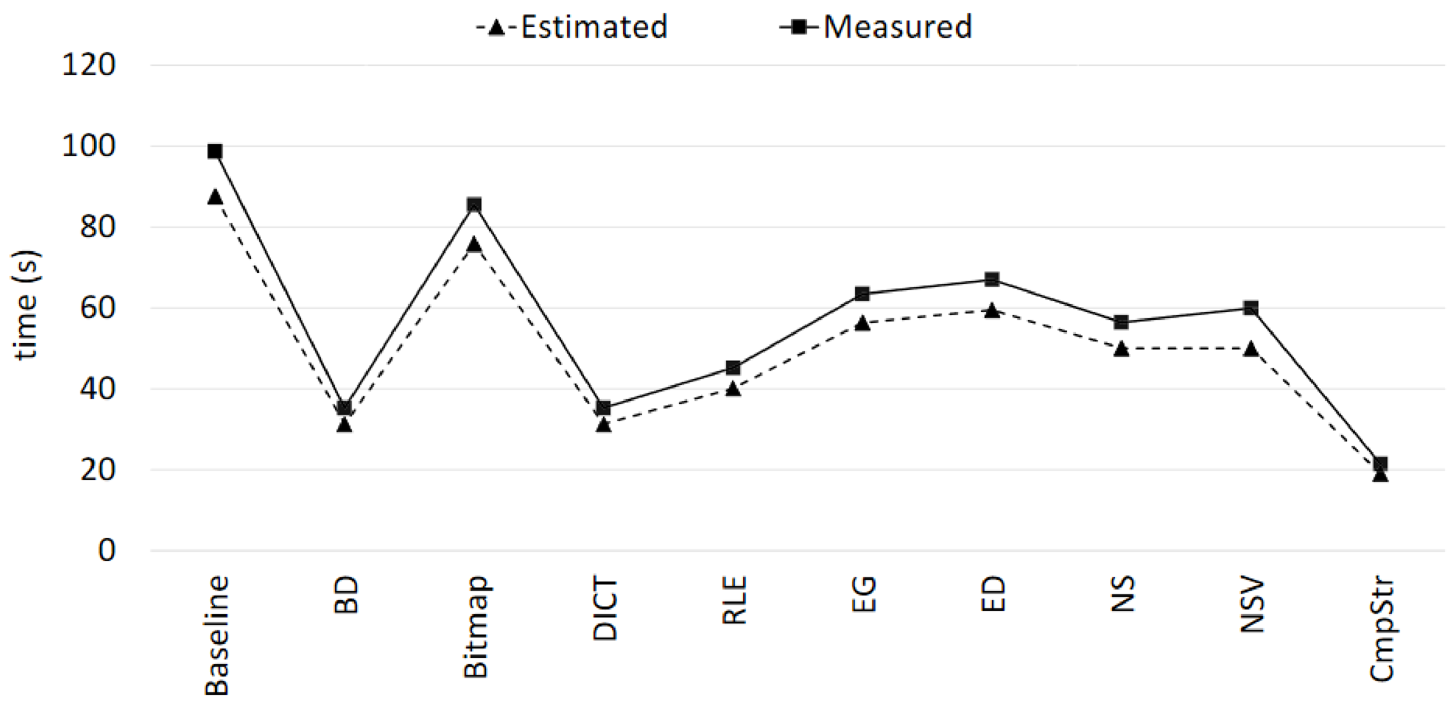


Fig. 9. Accuracy of the cost model. CmpStr is short for CompressStreamDB.

PURDUE UNIVERSITY

CompressStreamDB

- 3.24x throughput improvement
- 66.0% lower latency
- 66.8% space savings
- The system is positioned to integrate more compression schemes

Thank You

PURDUE
UNIVERSITY®