Blitzcrank: Fast Semantic Compression for In-Memory Online Transaction Processing

Yiming Qiao, Yihan Gao, Huanchen Zhang

Tsinghua University

In-Memory Compression Matters

➡ In-memory Databases are faster than On-disk Databases

Cache	Memory	SSD
1ns	☺ 80ns	10^5ns

In-Memory Compression Matters

➔ In-memory Databases are faster than On-disk Databases

Cache	Memory	SSD
1ns	☺ 80ns	10^5ns

➔ Memory is an expensive and limited resource.

Datacenter	Power	Hardware Cost
Memory Percentage	33.3%	37.1%

[ASPLOS'23, Meta]

Prior Work Focuses on Column Store

id	gender
1	M
2	M
3	M
4	M
5	F
6	M
7	F
8	M
9	M
10	M

Data of the same data type is stored together

Lightweight Encodings (e.g., RLE, FSST, LeCo)

Prior Work Focuses on Column Store

id	gender
1	M
2	M
3	M
4	M
5	F
6	M
7	F
8	M
9	M
10	M

Data of the same data type is stored together

Lightweight Encodings (e.g., RLE, FSST, LeCo)

Analytical workloads are read-mostly with large batched processing.

```
Select gender, Count(*)
  From user
 Group By gender
```

SIMD (e.g., Arrow, Parquet, FastLanes)

Row-Store Compression is the Missing Piece

id	gender	balance
1	M	19.8
2	M	22.5
3	M	19.8
4	M	10.5
5	F	6.2
6	M	8.4

Column-level Compression Does not Work

- Similar data is stored separately

Row-Store Compression is the Missing Piece

id	gender	balance
1	M	19.8
2	M	22.5
3	M	19.8
4	M	10.5
5	F	6.2
6	M	8.4

Column-level Compression Does not Work

Similar data is stored separately

A Tuple is Too Small to Compress

Compressor	25 bytes
ZSTD	1.1×

Row-Store Compression is the Missing Piece

id	gender	balance
1	M	19.8
2	M	22.5
3	M	19.8
4	M	10.5
5	F	6.2
6	M	8.4

Column-level Compression Does not Work

Similar data is stored separately

A Block is Large Enough to Compress

Compressor	25 bytes	100 KB
ZSTD	1.1×	9.6×

Block Compressor Has High Access Latency

id	gender	balance
1	M	19.8
2	M	22.5
3	M	19.8
4	M	10.5
5	F	6.2
6	M	8.4

- ☺ Compression Factor
- ☺ Compression Throughput

Block Compressor Has High Access Latency

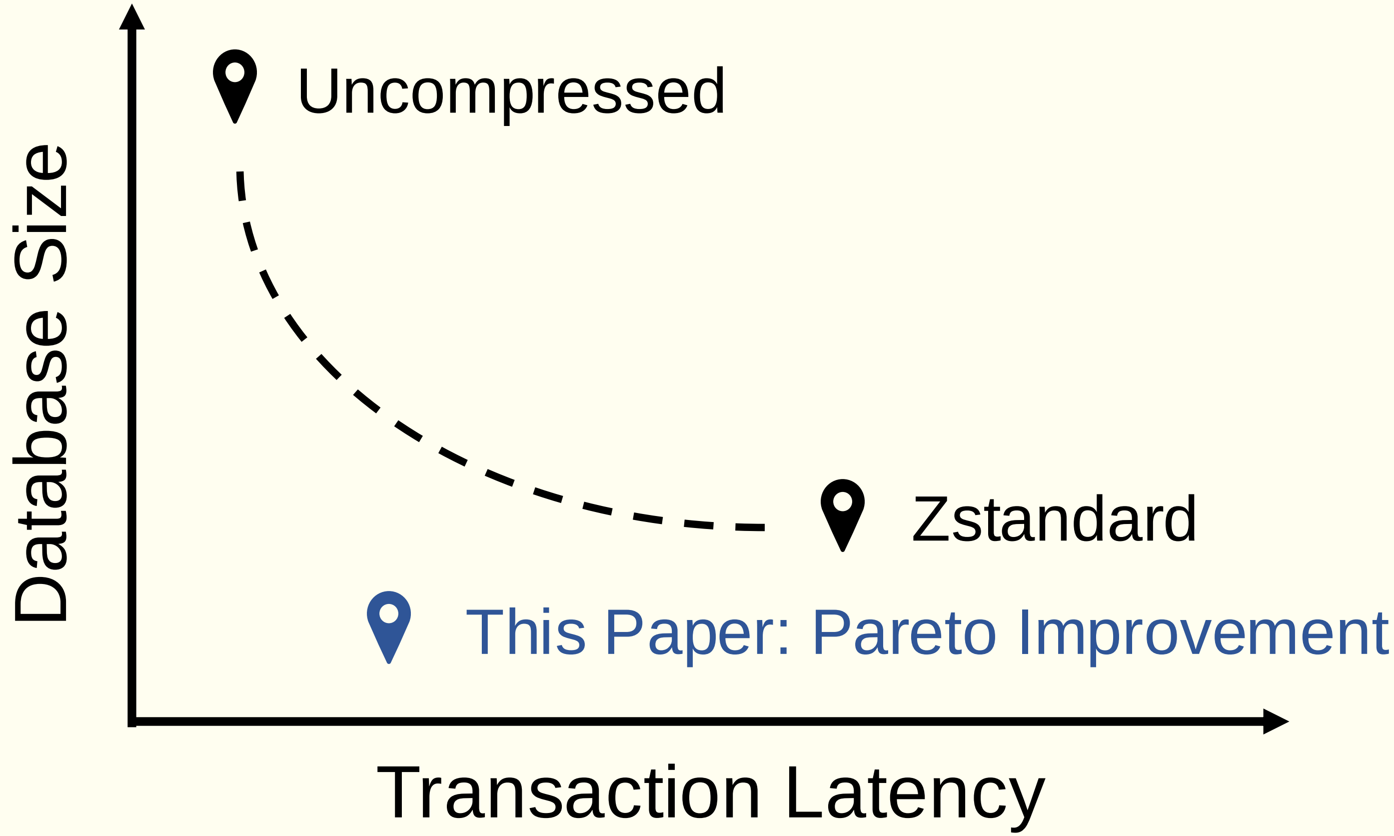
id	gender	balance
1	M	19.8
2	M	22.5
3	M	19.8
4	M	10.5
5	F	6.2
6	M	8.4

- ☺ Compression Factor
- ☺ Compression Throughput
- ☹ Tuple Random-access Latency

Block Compressor (e.g., ZSTD) must decompress the entire compression block to access a single tuple

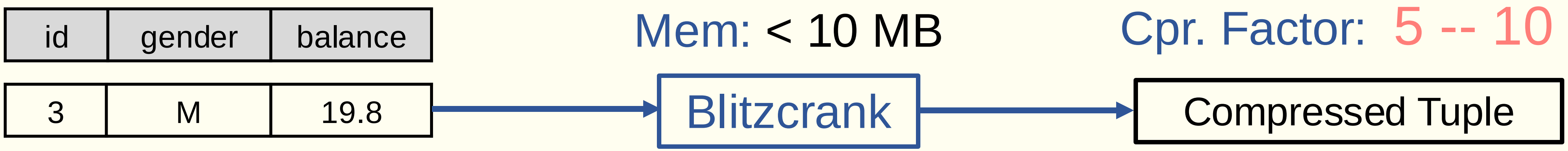
This Paper Offers a Tuple-level Compressor

A stand-alone C++ library for compressing row-store OLTP databases



This Paper Offers a Tuple-level Compressor

A stand-alone C++ library for compressing row-store OLTP databases



	Compression	Decompression
Latency:	2 us/tuple	1 us/tuple
Throughput:	100 MB/s	200 MB/s

Compression = Modeling + Encoding

Find the features of the uncompressed data

id	gender	name
1	F	Taylor
2	M	Alex
3	F	Alex
4	F	Taylor
5	F	Taylor

Compression = **Modeling** + Encoding

Find the features of the uncompressed data

id	gender	name
1	F	Taylor
2	M	Alex
3	F	Alex
4	F	Taylor
5	F	Taylor

Zstandard treats the uncompressed data simply as consecutive bytes

- ☺ Make sense for General-purpose Compressors
- ☹ High-level semantics are lost (e.g. table schema)

Compression = **Modeling** + Encoding

Find the features of the uncompressed data

id	gender	name
1	F	Taylor
2	M	Alex
3	F	Alex
4	F	Taylor
5	F	Taylor

Zstandard treats the uncompressed data simply as consecutive bytes

- Make sense for General-purpose Compressors
- High-level semantics are lost (e.g. table schema)

Blitzcrank uses the Semantic Modeling

Compression = **Modeling** + Encoding

Find the features of the uncompressed data

id	gender	name
1	F	Taylor
2	M	Alex
3	F	Alex
4	F	Taylor
5	F	Taylor

Highly skewed distribution for attribute values

e.g., few users are male:

P (gender = M) = 0.2

Compression = **Modeling** + Encoding

Find the features of the uncompressed data

id	gender	name
1	F	Taylor
2	M	Alex
3	F	Alex
4	F	Taylor
5	F	Taylor

Highly skewed distribution for attribute values

e.g., few users are male:

P (gender = M) = 0.2

Correlation between attributes of the same tuple

e.g., all Taylors are female:

P (gender = F | name = Taylor) = 1

Compression = **Modeling** + Encoding

Find the features of the uncompressed data

id	gender	name
1	F	Taylor
2	M	Alex
3	F	Alex
4	F	Taylor
5	F	Taylor

Semantic Model for name

P (name = Alex) = 0.4

P (name = Taylor) = 0.6

Semantic Model for gender

P (gender = F | name = Taylor) = 1

P (gender = F | name = Alex) = 0.5

Compression = **Modeling** + Encoding

Find the features of the uncompressed data

id	gender	name
1	F	Taylor
2	M	Alex
3	F	Alex
4	F	Taylor
5	F	Taylor

Semantic Model for name

P (name = Alex) = 0.4

P (name = Taylor) = 0.6

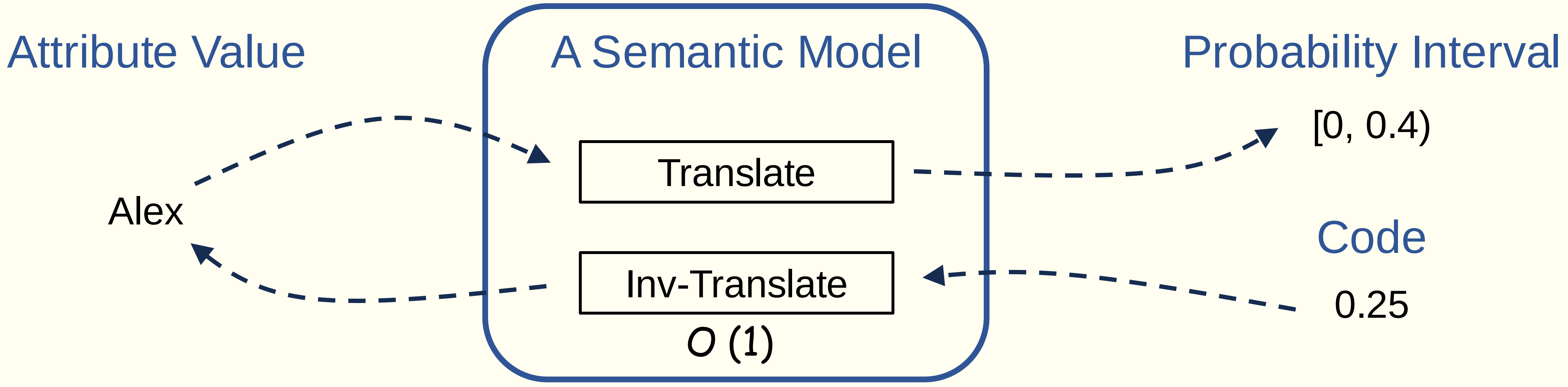
Semantic Model for gender

P (gender = F | name = Taylor) = 1

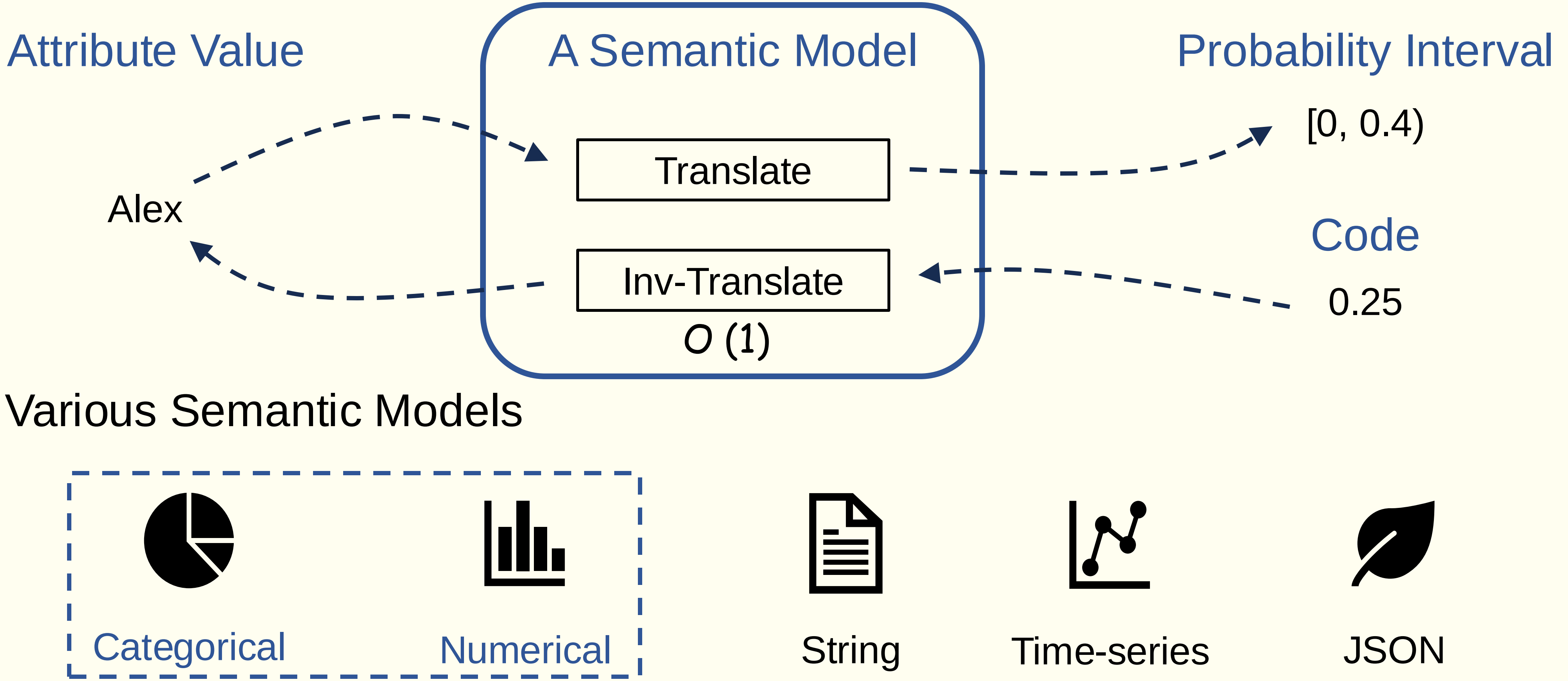
P (gender = F | name = Alex) = 0.5

P (gender, name) = P (name) × P (gender | name)

Semantic Models in Blitzcrank

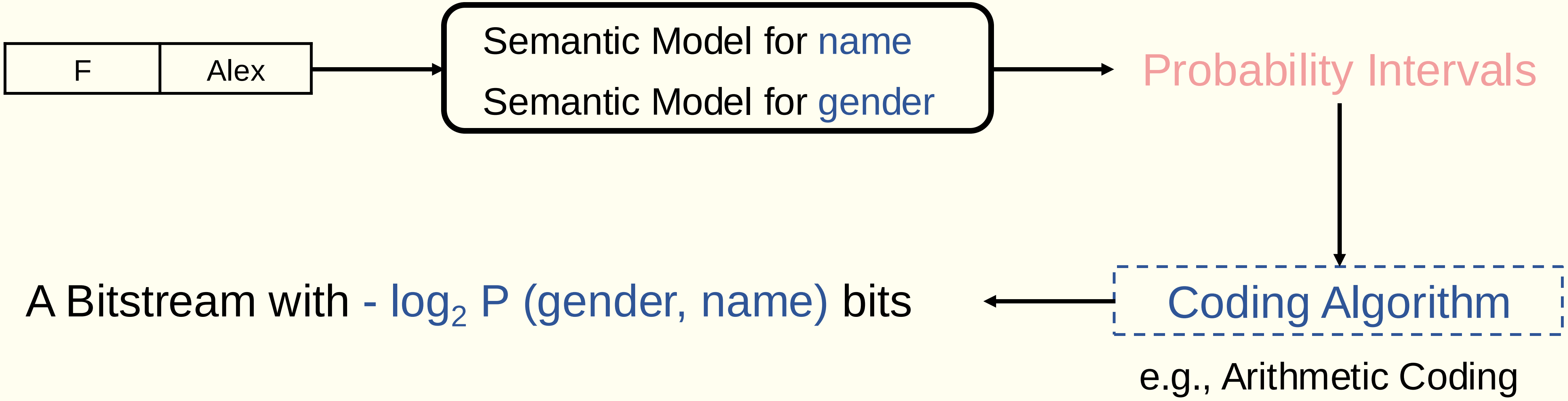


Semantic Models in Blitzcrank



Compression = Modeling + **Encoding**

Encode the data using learned semantic models



Encoding of Arithmetic Coding

Semantic Model for name

Semantic Model for gender

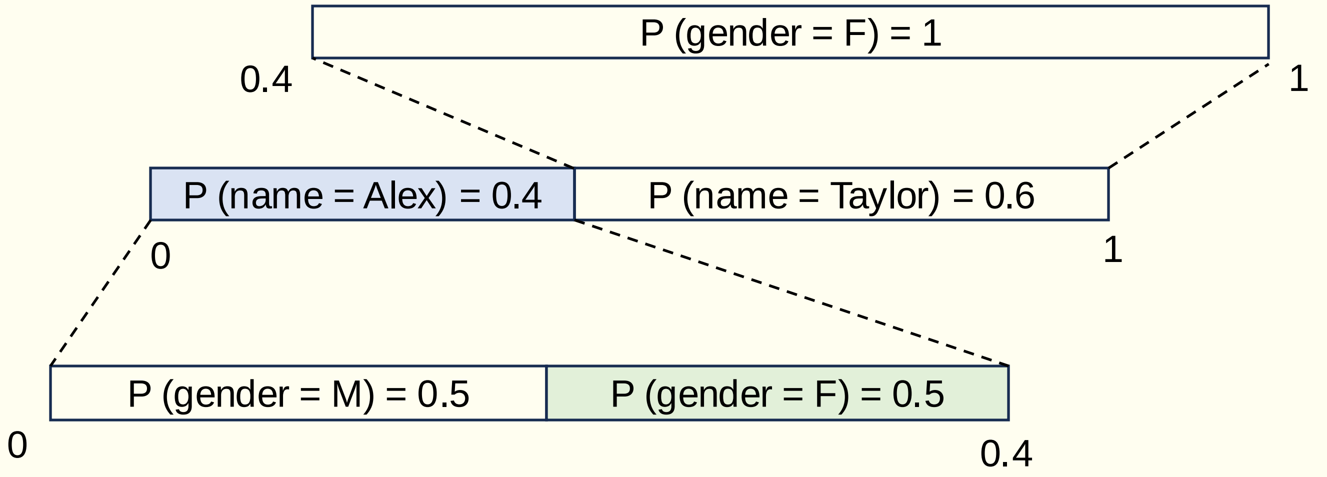
gender	name
F	Alex

→ [0, 0.4) [0.5, 1)

Encoding of Arithmetic Coding

Semantic Model for name

Semantic Model for gender

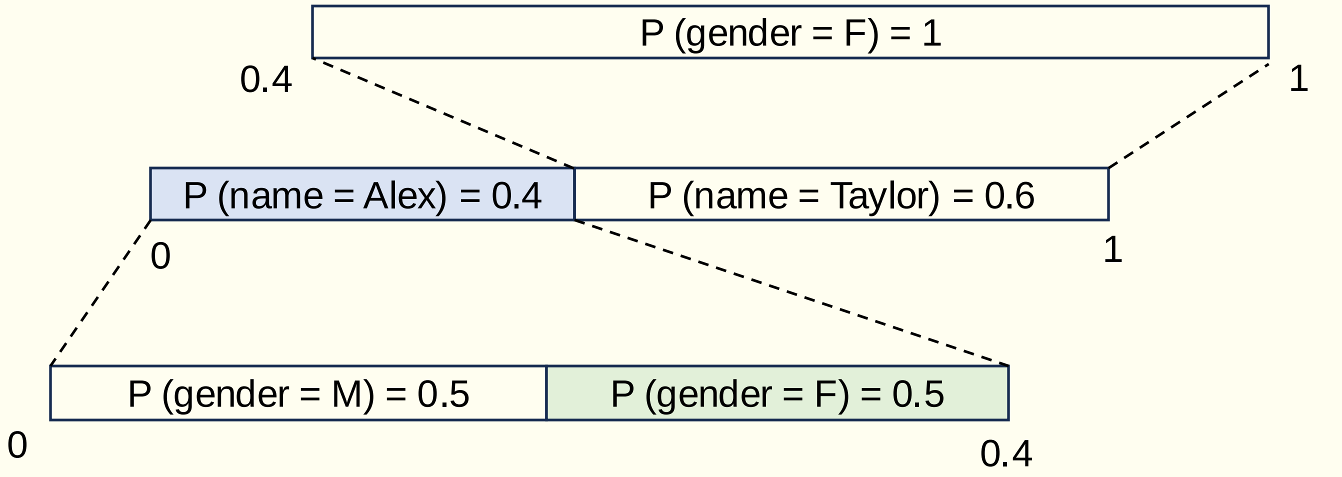


gender	name
F	Alex

$\rightarrow$ [0, 0.4) $\otimes$ [0.5, 1) $\rightarrow$ [0.2, 0.4)

Numerous floating-point calculations

Encoding of Arithmetic Coding



Semantic Model for name

Semantic Model for gender

gender	name
F	Alex

→ $[0, 0.4) \otimes [0.5, 1)$ → $[0.2, 0.4)$ → $(.010)_2$

Numerous floating-point calculations

Arithmetic Coding vs. Delayed Coding

Floating-point Calculation

[0, 0.4) ⊗ [0.5, 1)

Simple Integer Probability

e.g., 4-bit integer [0, 6) [8, 16)

Variable-length Code

[0, 0.4) ⟶ $(.00)_2$

[0.5, 1) ⟶ $(.1)_2$

Fixed-length Code

with near-entropy performance

Options of Fixed-length Codes

gender	name	age
F	Tom	21

→ [1/16, 1/4) [1/8, 1/2) [3/16, 1/4)

Options of Fixed-length Codes

gender	name	age
F	Tom	21

→

~~[1/16, 1/4)~~ ~~[1/8, 1/2)~~ ~~[3/16, 1/4)~~

[1, 4) [2, 8) [3, 4)

Options of Fixed-length Codes

gender	name	age
F	Tom	21

→ ~~[1/16, 1/4)~~ ~~[1/8, 1/2)~~ ~~[3/16, 1/4)~~

[1, 4) [2, 8) [3, 4)

For an interval [L, R), any 4-bit integer in this interval can be used as the code

Fixed-length (4-bit) Code	0001	0010	0011

Options of Fixed-length Codes

gender	name	age
F	Tom	21

→ ~~[1/16, 1/4)~~ ~~[1/8, 1/2)~~ ~~[3/16, 1/4)~~

[1, 4) [2, 8) [3, 4)

For an interval [L, R), any 4-bit integer in this interval can be used as the code

Fixed-length (4-bit) Code	0001	0010	0011
Bitstream	$(0001\ 0010\ 1001)_2$		
Interval Entropy in bits	2.4	1.4	4

☹ 12 bits:7.8 bits, Waste Many Bits

Code Selection itself Carries Information

[1, 4) [2, 8) [3, 4)

We have 3 code options for an interval [1, 4)

1, 2, 3

We have 6 code options for an interval [2, 8)

2, 3, 4, 5, 6, 7

Code 1	Code 2	State
1	2	0
1	3	1
…	…	…
3	6	16
3	7	17

Code Selection itself Carries Information

[1, 4) [2, 8) [3, 4)

We have 3 code options for an interval [1, 4)

1, 2, 3

We have 6 code options for an interval [2, 8)

2, 3, 4, 5, 6, 7

Code 1	Code 2	State
1	2	0
1	3	1
…	…	…
3	6	16
3	7	17

We can use the first two intervals to represent the third interval

Offer 18 states

Require 16 states

Encode Three Intervals Using Two 4-bits

The first two intervals form a 2-digit mixed-radix (3, 6) numeral system

	[1, 4)	[2, 8)	[3, 4)	
Radix	3	6		$N = a_3 b_6 = a \times 6 + b$

Encode Three Intervals Using Two 4-bits

The first two intervals form a 2-digit mixed-radix (3, 6) numeral system

	[1, 4)	[2, 8)	[3, 4)
Radix	3	6	
Initial Code	0001	0010	0011 (3)
Digits (a, b)	0	3	

$N = a_3 b_6 = a \times 6 + b$

$3 = 0_3 3_6 = \underline{0} \times 6 + \underline{3}$

Encode Three Intervals Using Two 4-bits

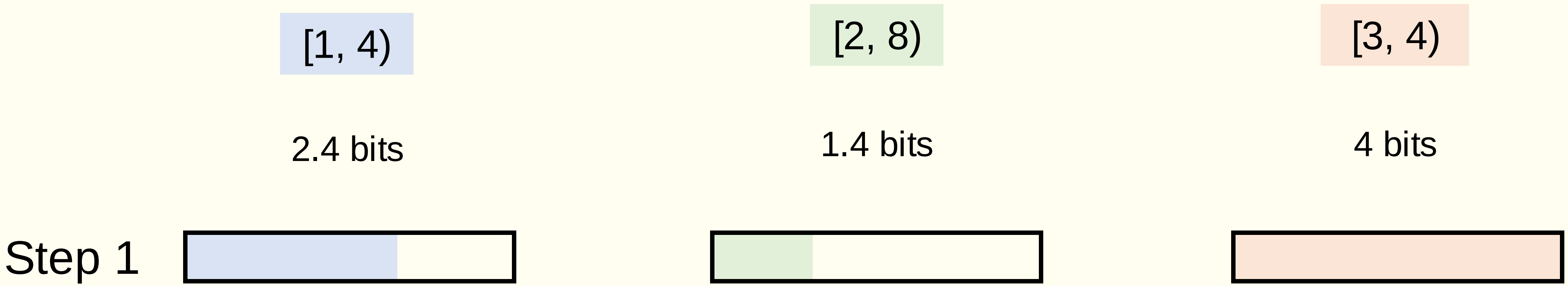
The first two intervals form a 2-digit mixed-radix (3, 6) numeral system

	[1, 4)	[2, 8)	[3, 4)
Radix	3	6	
Initial Code	0001	0010	0011 (3)
Digits (a, b)	0	3	
Selected Code	1 + 0	2 + 3	
Resulting Bitstream	(0001	$0101)_2$	

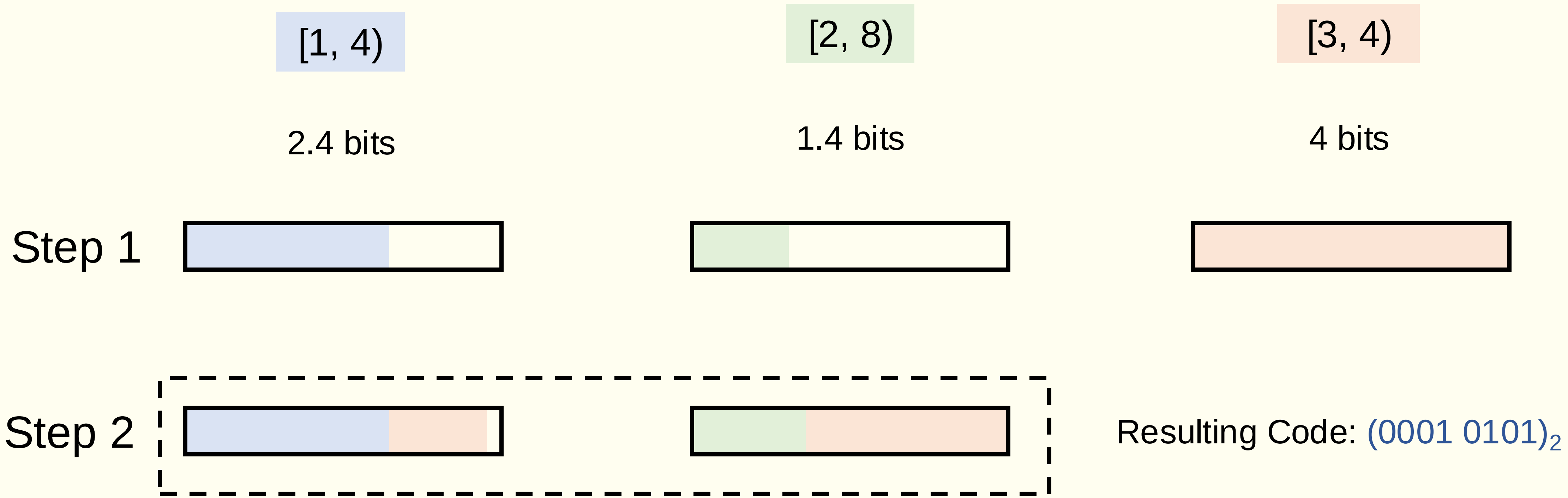
$$N = a_3 b_6 = a \times 6 + b$$

$$3 = 0_3 3_6 = \underline{0} \times 6 + \underline{3}$$

Encoding of Delayed Coding



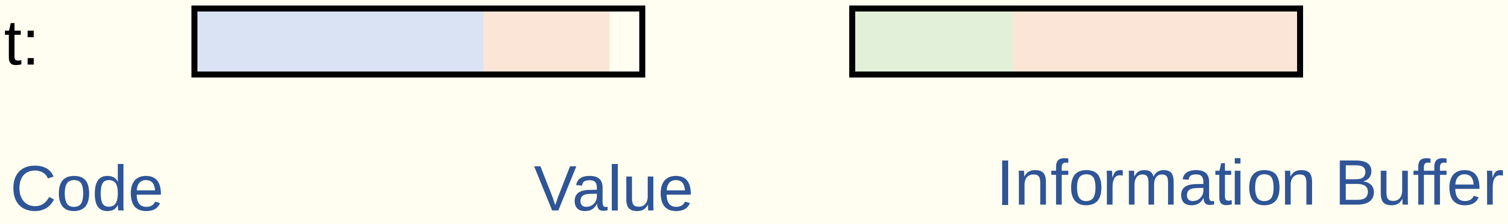
Encoding of Delayed Coding



It uses 8 bits to represent three intervals, with a total entropy of 7.8 bits

Decoding of Delayed Coding

Decoding Input:



Step 1

Step 2

Step 2

Decoding of Delayed Coding

Decoding Input:

	Code	Value	Information Buffer
Step 1			
Step 2			
Step 3			

Decoding of Delayed Coding

Decoding Input:

	Code	Value	Information Buffer
Step 1		⟹ name = Tom	
Step 2			
Step 3			

Decoding of Delayed Coding

Decoding Input:

	Code	Value	Information Buffer
Step 1		$\Longrightarrow$ name = Tom	
Step 2		$\Longrightarrow$ gender = F	
Step 3			

Decoding of Delayed Coding

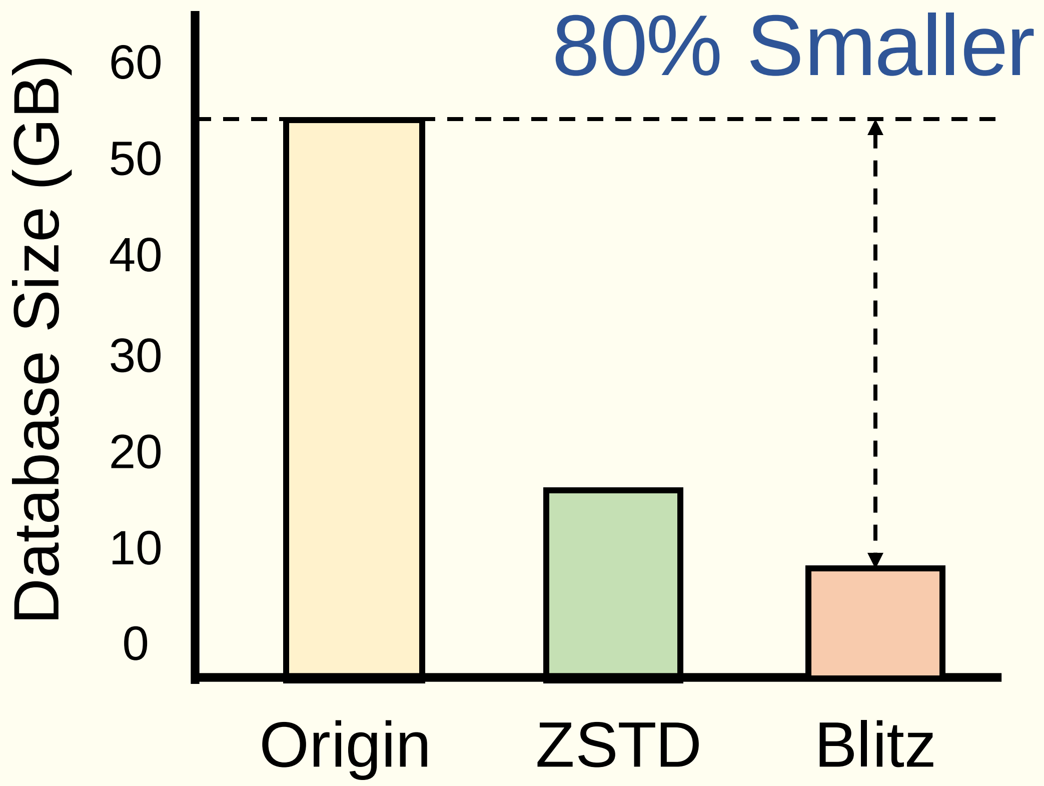
Decoding Input:

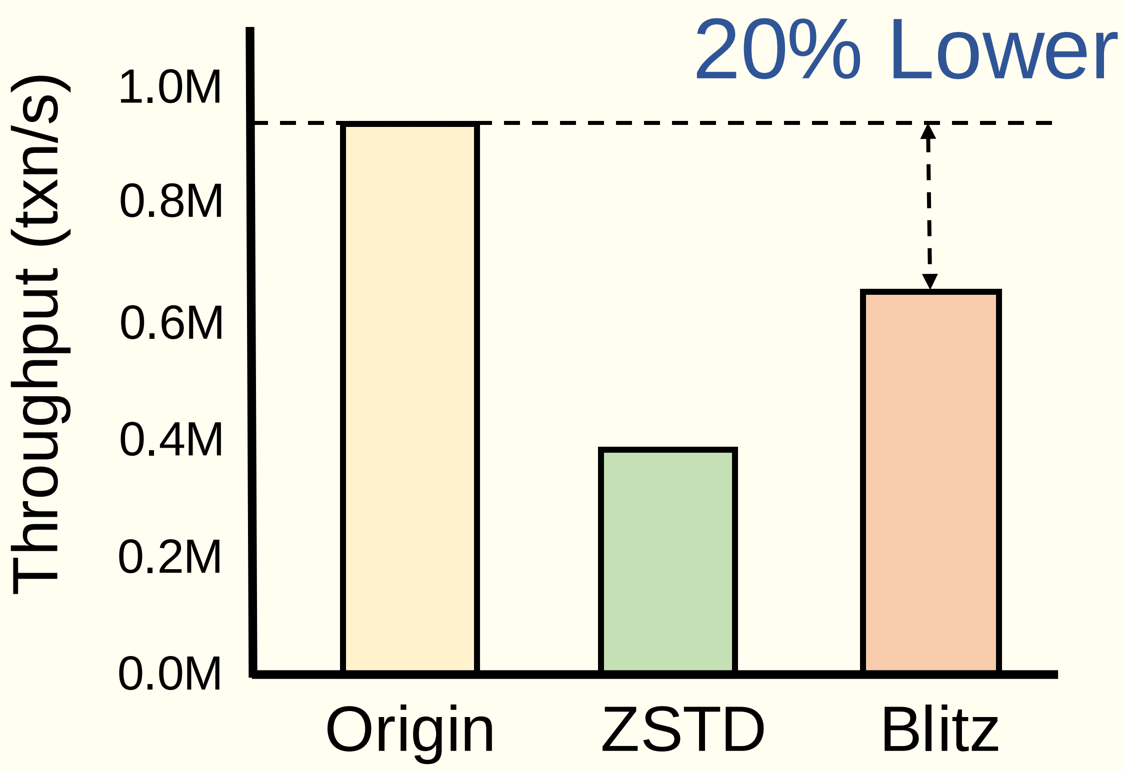
	Code	Value	Information Buffer
Step 1		⟹ name = Tom	
Step 2		⟹ gender = F	
Step 3		⟹ age = 21	

Decoding Result:

F	Tom	21

Applying Entropy Coding to Real Systems?





OLTP Compression Takeaways

- Modern Entropy Coding is very Fast
- Compression Granularity is the Key Factor for OLTP
- Source: https://github.com/YimingQiao/Blitzcrank