Кодеки для Time-Series в ClickHouse

Василий Немков @ Altinity

11.07.2019

Time-Series

- Что такое Time-Series ?
  - Последующее и предыдущее похожи (обычно)
- Примеры
  - Показания датчиков
  - Тикеры биржевых торгов
- Данных много
- ClickHouse уже используется как Time-Series база данных и показывает хорошие результаты [1]

Хранение данных

- кодеки превращают данные из колонки в байты на HDD/SDD
  - кодек может быть свой у каждой колонки
  - кодеки можно группировать
- пример данных и их представления
- соотношение сжатие\скорость

Кодеки

- изначально - только LZ4 и ZSTD
- добавлен фреймворк для создания своих кодеков (19.1.6) [2]
  - легко (относительно) можно добавить свой кодек
- список кодеков на текущий момент:
  - NONE
  - LZ4
  - ZSTD
  - Delta
  - T64
  - DoubleDelta
  - Gorilla

Кодеки

```
CREATE TABLE codec_test1_seq (
 n Int32,
 n32                  Int32 Codec(NONE),
 n32_doubledelta      Int32 Codec(DoubleDelta),
 l_n32_doubledelta Int32 Codec(DoubleDelta, LZ4),
 l_n32_gorilla        Int32 Codec(Gorilla, LZ4)
) Engine = MergeTree
PARTITION BY tuple() ORDER BY tuple();
```

DoubleDelta

- Реализация основана на статье от FB:
  - Gorilla: A Fast, Scalable, In-Memory Time Series Database [3]
- Работает для всех целочисленных типов (и типов представимых) [4]
  - Int{8,16,32,64}
  - UInt{8,16,32,64}
  - Date
  - DateTime

DoubleDelta

1. Delta

$$d_i = v_i - v_{i-1}$$

2. Delta of Delta

$$dd_i = d_i - d_{i-1}$$

3. Компактное бинарное представление

DoubleDelta

dd_i	0	(-63, 64)	(-255, 256)	(-2047, 2048)	(-2147483648, 2147483647)	...
Бинарный префикс	0	10	110	1110	11110	11111
Количество бит данных	0	7	9	12	32	64
Итого бит	1	9	12	16	37	69

DoubleDelta

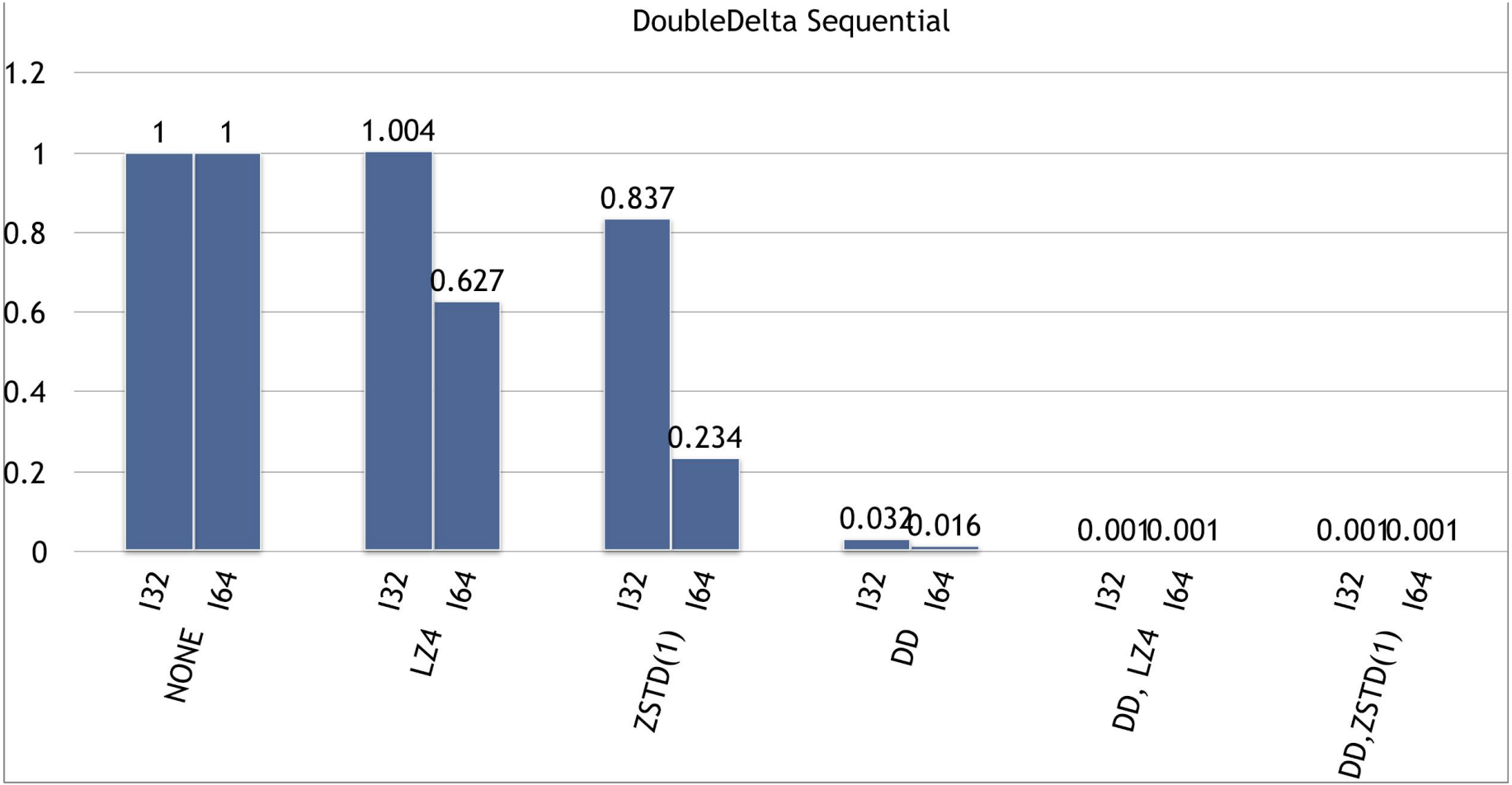
Data	5	10	15	20	25
Delta	-	5	5	5	5
DD	-	-	0	0	0
Output	00000101 (8bit)	00000101 (8bit)	0 (1bit)	0 (1bit)	0 (1bit)

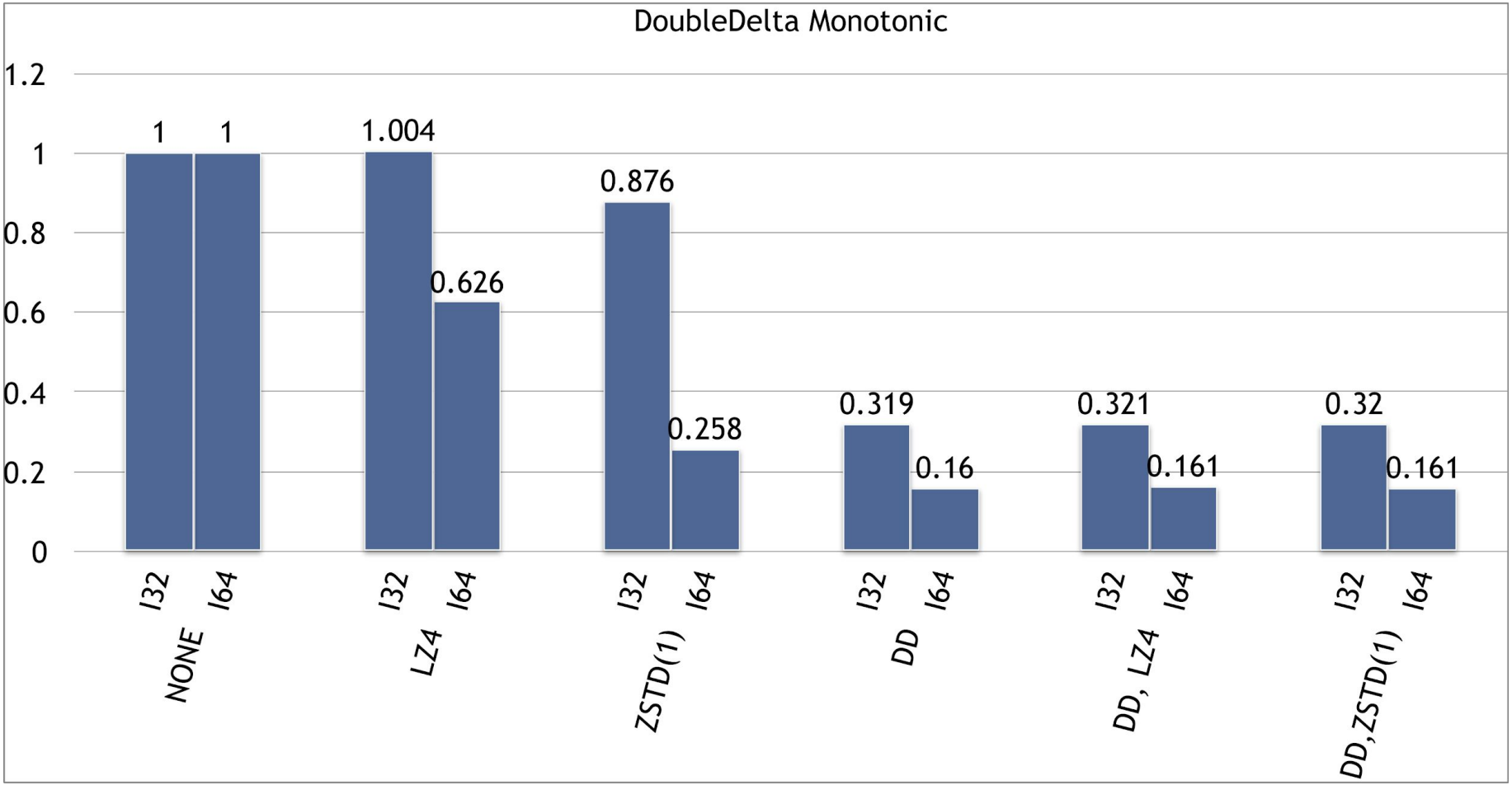
DoubleDelta

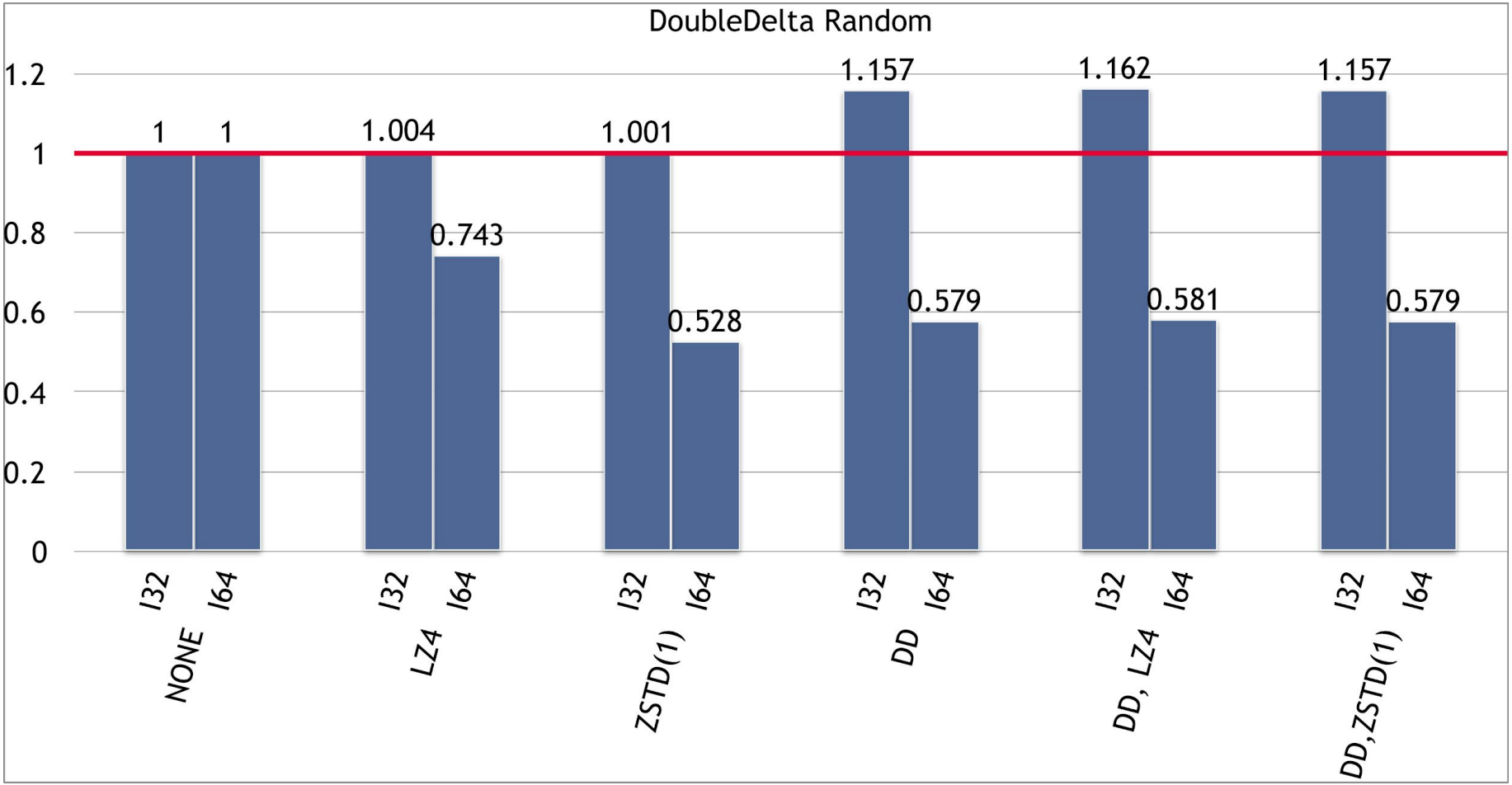
Data	1	2	4	8	16
Delta	-	1	2	4	8
DD	-	-	1	2	4
Output	00000001 (8bit)	00000001 (8bit)	10 0000001 (9bit)	10 0000010 (9bit)	10 0000100 (9bit)

DoubleDelta

- Компрессия [5]
- Меньше – лучше (если вы не производитель HDD)
- Данные
  - Последовательные (постоянный шаг)
  - Монотонные (переменный шаг)
  - Случайные (до 1 Миллиарда)


DoubleDelta Sequential
1.2
1
0.8
0.6
0.4
0.2
0
1
1
1.004
0.627
0.837
0.234
0.032
0.016
0.001
0.001
0.001
0.001
I32
I64
NONE
LZ4
ZSTD(1)
DD
DD, LZ4
DD,ZSTD(1)


DoubleDelta Monotonic
1.2
1
0.8
0.6
0.4
0.2
0
1
1
1.004
0.626
0.876
0.258
0.319
0.16
0.321
0.161
0.32
0.161
I32
I64
NONE
LZ4
ZSTD(1)
DD
DD, LZ4
DD,ZSTD(1)


DoubleDelta Random
1.2
1
0.8
0.6
0.4
0.2
0
1
1
1.004
0.743
1.001
0.528
1.157
0.579
1.162
0.581
1.157
0.579
I32
I64
NONE
LZ4
ZSTD(1)
DD
DD, LZ4
DD,ZSTD(1)

Gorilla

- Реализация основана на статье от FB:
  - Gorilla: A Fast, Scalable, In-Memory Time Series Database [3]
- Работает для всех 32 и 64 битных типов [4]
  - Int{32,64}
  - UInt{32,64}
  - Float{32,64}
  - Date
  - DateTime
- Разработан для типов с плавающей точкой

Gorilla

1. BIT-XOR

$$g_i = v_i \oplus v_{i-1}$$

2. Компактное бинарное представление

Gorilla

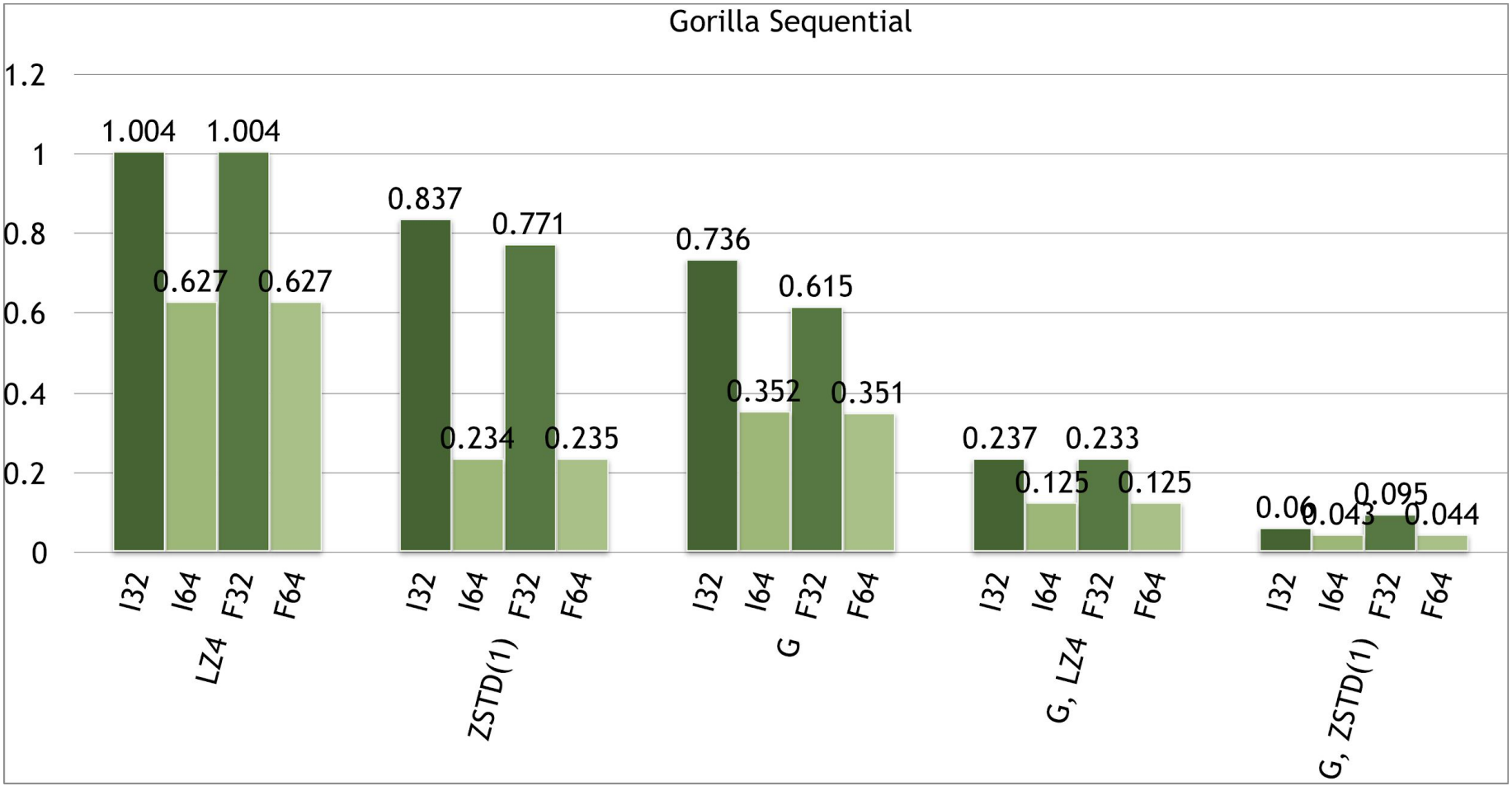
v_i	01000000001011011111100001010100
v_{i-1}	01000000000000100101011100001000
g_i	00000000001011001101001111011100
	lzb=10 data bits=20

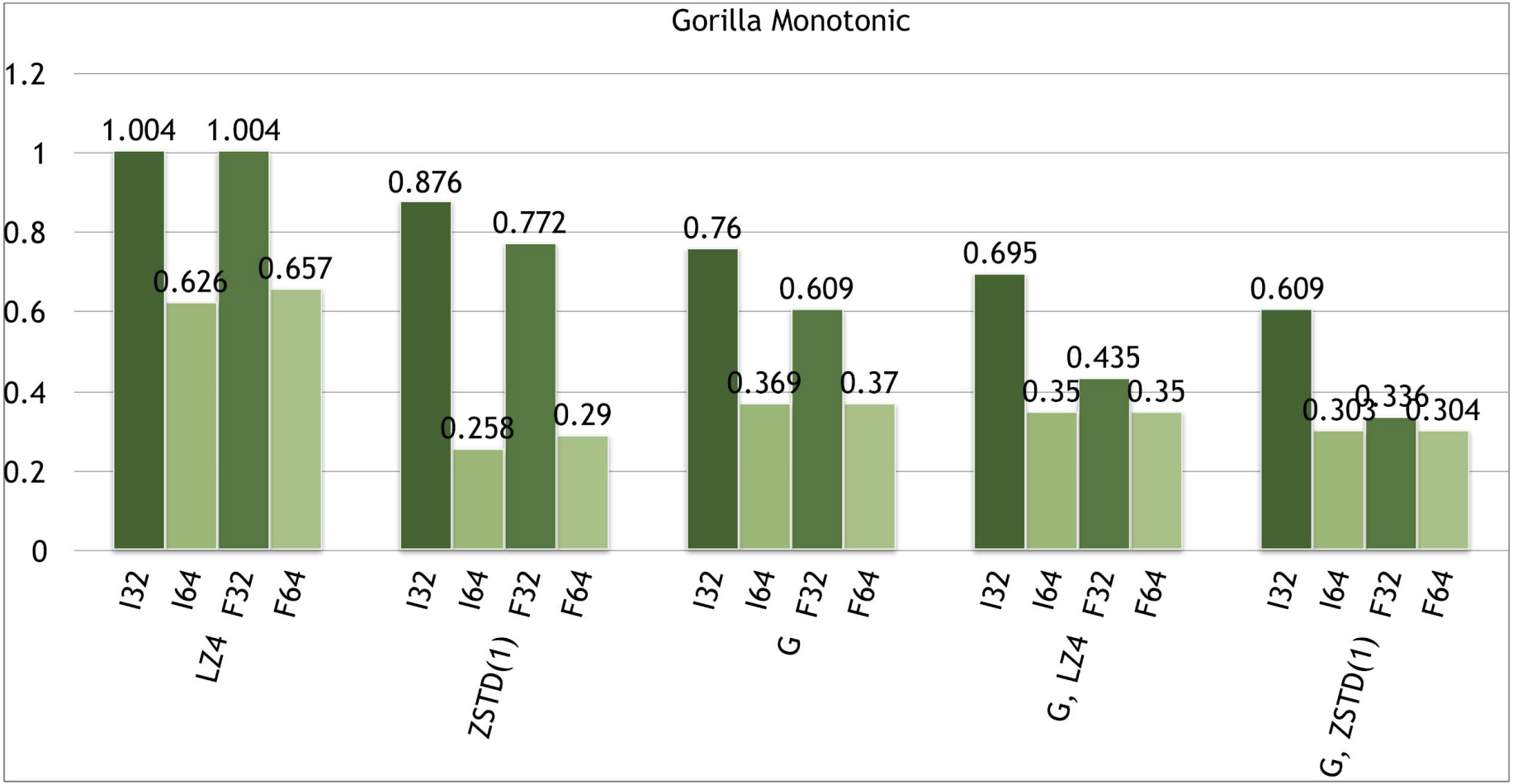
Gorilla

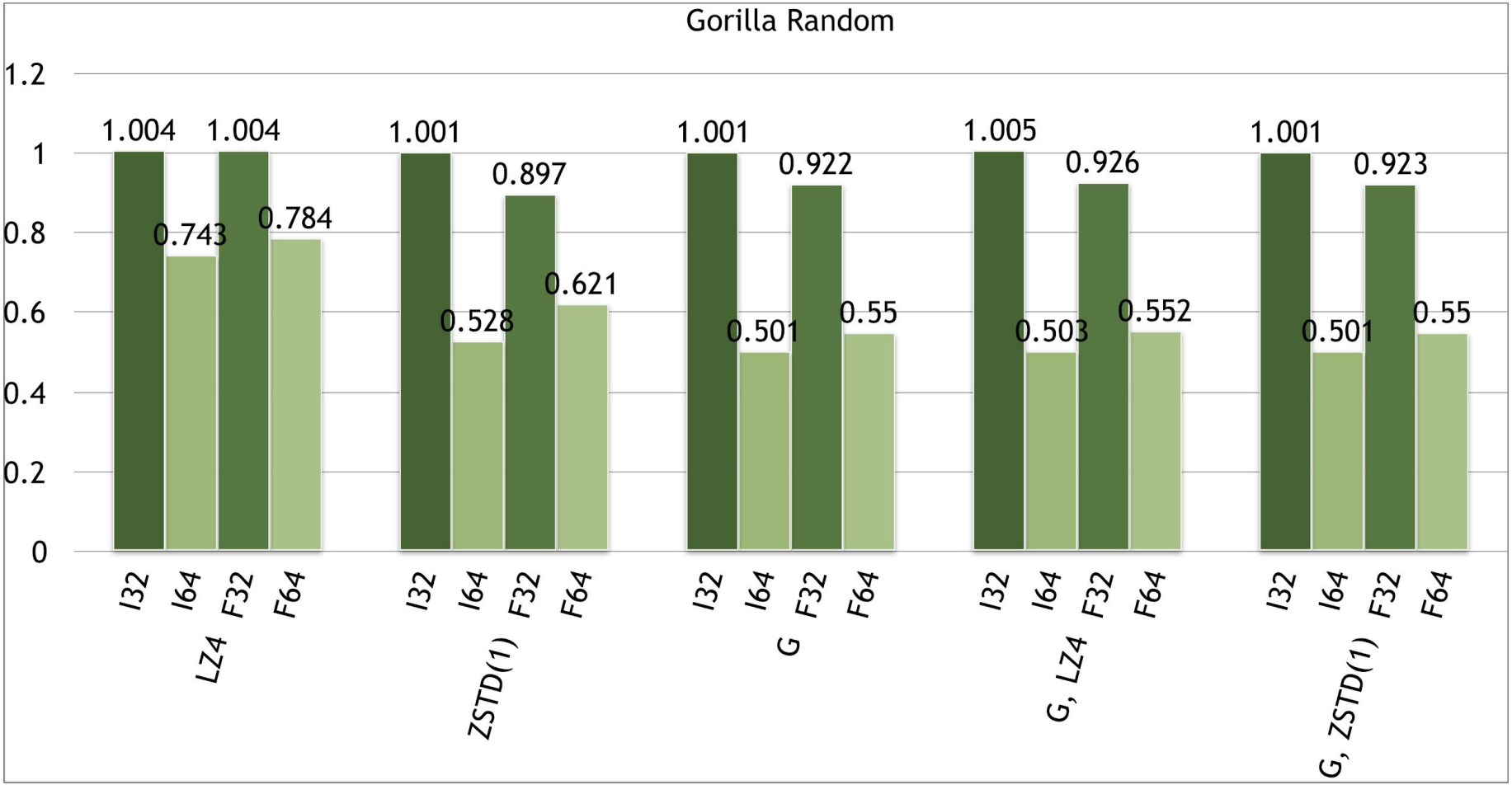
g_i	0	lzb и db в диапазоне g_{i-1}	...
Бинарный префикс	0	10	11
Количество бит данных	0	? (20)	5 of lzb 6 of db 20 of data
Итого бит	1	22	33

Gorilla

- Компрессия [5]
- Меньше – лучше
- Данные (те же самые что и для DoubleDelta)
  - Последовательные (постоянный шаг)
  - Монотонные (переменный шаг)
  - Случайные (до 1 Миллиарда)


Gorilla Sequential
1.2
1
0.8
0.6
0.4
0.2
0
1.004
1.004
0.627
0.627
0.837
0.771
0.234
0.235
0.736
0.615
0.352
0.351
0.237
0.233
0.125
0.125
0.06
0.043
0.095
0.044
I32
I64
F32
F64
LZ4
ZSTD(1)
G
G, LZ4
G, ZSTD(1)

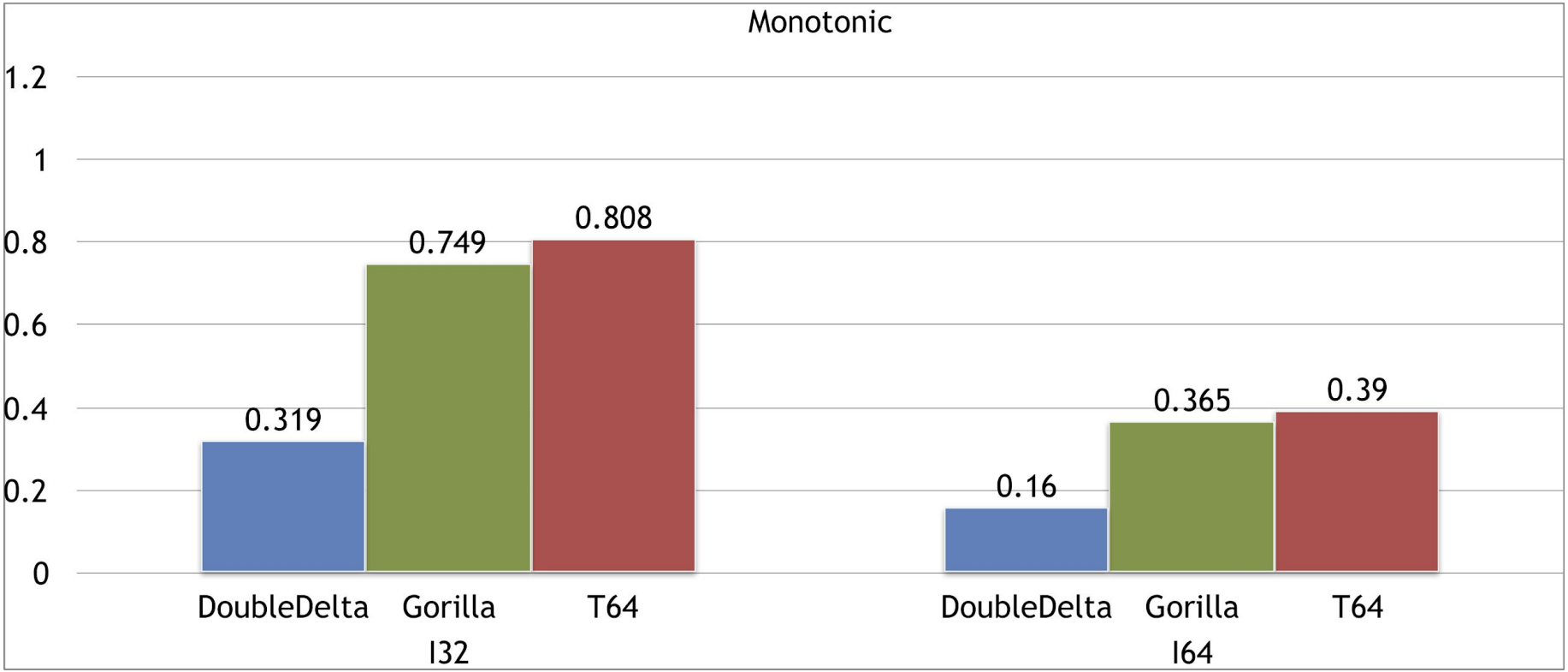

Gorilla Monotonic
1.2
1
0.8
0.6
0.4
0.2
0
1.004
0.626
1.004
0.657
0.876
0.258
0.772
0.29
0.76
0.369
0.609
0.37
0.695
0.35
0.435
0.35
0.609
0.303
0.336
0.304
I32
I64
F32
F64
LZ4
ZSTD(1)
G
G, LZ4
G, ZSTD(1)


Gorilla Random
1.2
1
0.8
0.6
0.4
0.2
0
1.004
0.743
1.004
0.784
1.001
0.528
0.897
0.621
1.001
0.501
0.922
0.55
1.005
0.503
0.926
0.552
1.001
0.501
0.923
0.55
I32
I64
F32
F64
LZ4
ZSTD(1)
G
G, LZ4
G, ZSTD(1)

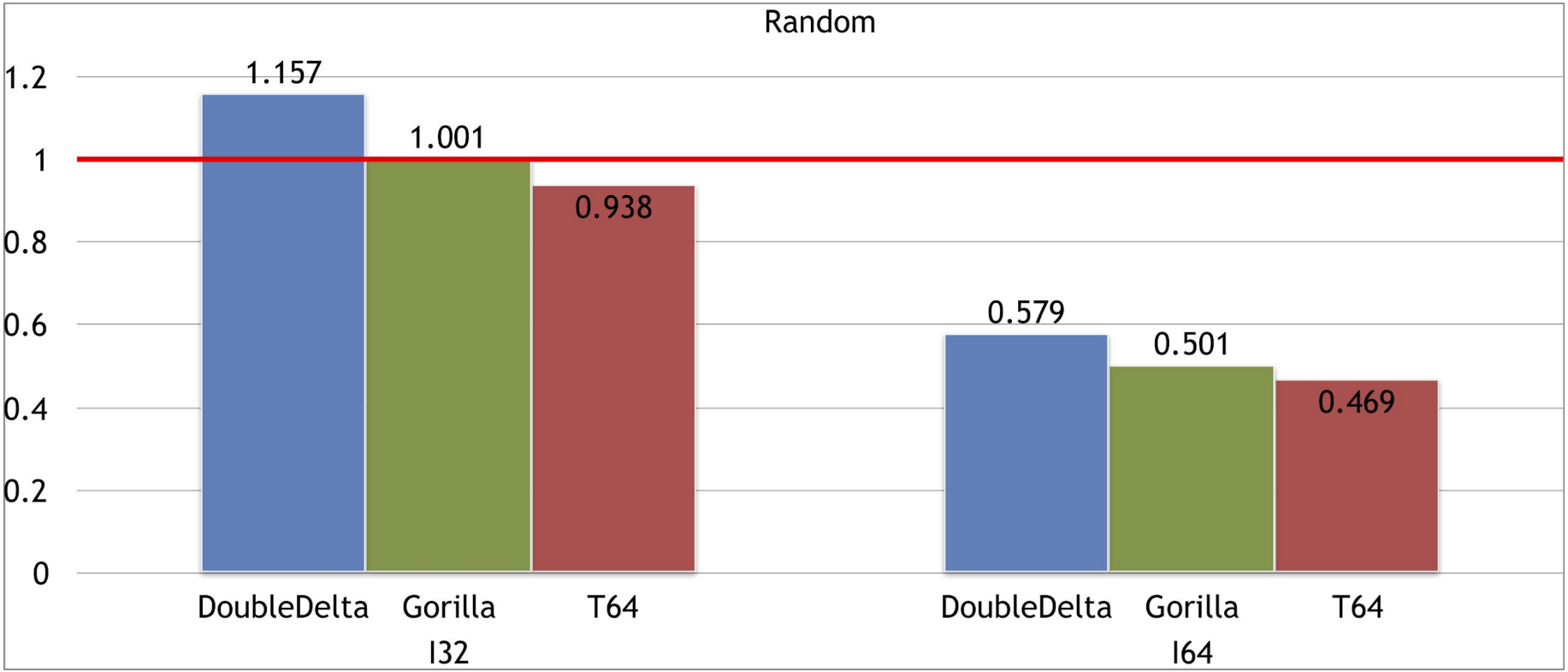
Сравнение кодеков между собой

- T64, DoubleDelta, Gorilla
- Только для Int32 и Int64
- Данные
  - Монотонные (переменный шаг)
  - Случайные (до 1 Миллиарда)

Сравнение кодеков между собой



Сравнение кодеков между собой



Сравнение кодеков между собой

Выводы

- DoubleDelta для целых значений TS
- Gorilla для дробных значений TS
- T64 для произвольных значений

Спасибо!

BTW, we are hiring...
Вопросы?

Ссылки

1. https://www.altinity.com/blog/clickhouse-for-time-series (CH for Time-Series data)
2. https://github.com/yandex/ClickHouse/pull/3899 (Custom compression codecs PR)
3. http://www.vldb.org/pvldb/vol8/p1816-teller.pdf (Gorilla & DoubleDelta paper)
4. https://github.com/yandex/ClickHouse/pull/5600 (Gorilla & DoubleDelta PR)
5. https://www.altinity.com/blog/2019/7/new-encodings-to-improve-clickhouse (New Encodings to Improve ClickHouse Efficiency)