NOSQL INSIDE SQL

STRATEGY AND TACTICS

DMITRY DOLGOV

06-07-2017

Не отвлекайся на всякий вздор

Только Postgres
Только хардкор

zalando

zalando

➔ Jsonb internals and performance-related factors

- ➔ Jsonb internals and performance-related factors
- ➔ Benchmarks

- ➔ Jsonb internals and performance-related factors
- ➔ Benchmarks
- ➔ How to shoot yourself in the foot

Internals

Performance-related factors

Performance-related factors

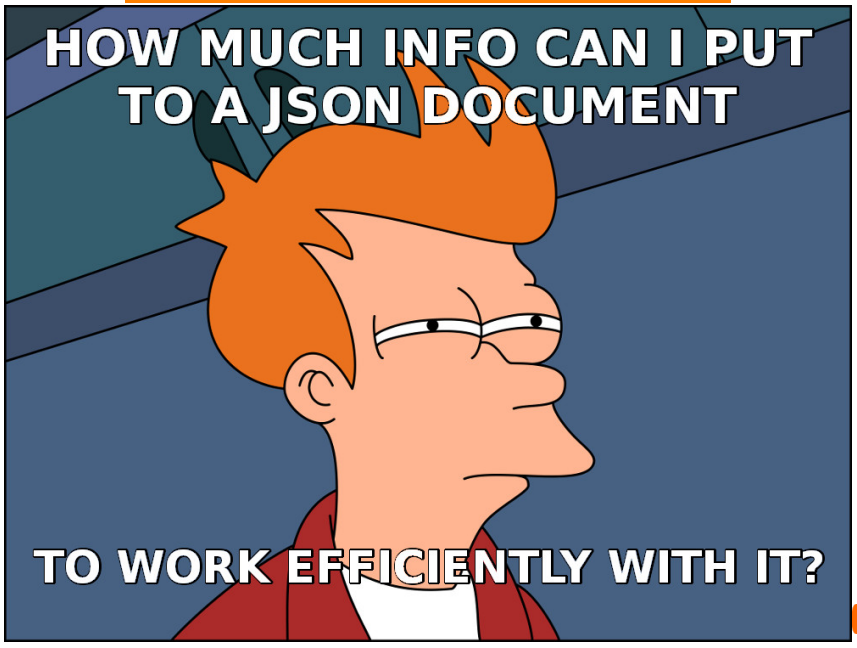
➔ On-disk representation

Performance-related factors

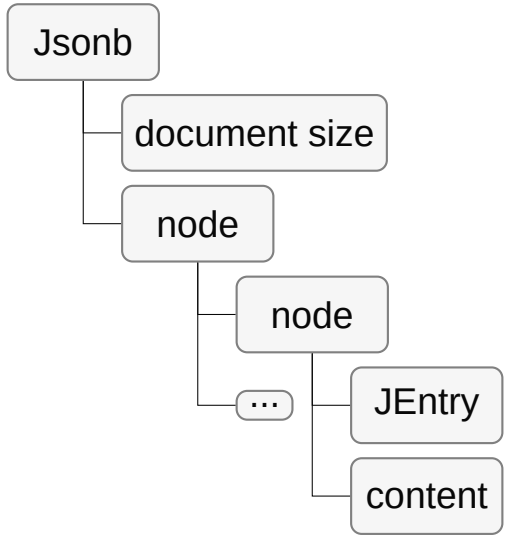
- ➔ On-disk representation
- ➔ In-memory representation

Performance-related factors

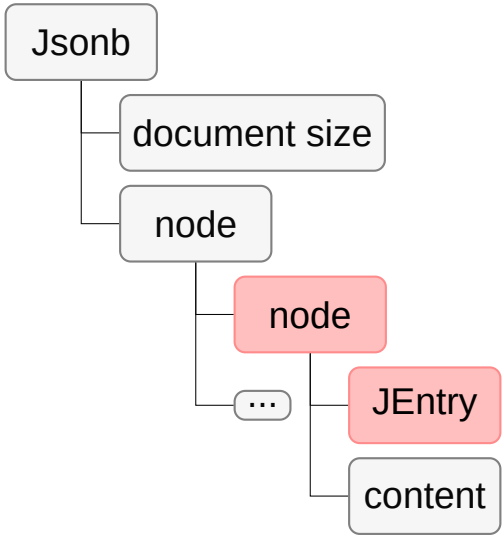
- ➔ On-disk representation
- ➔ In-memory representation
- ➔ Indexing support


HOW MUCH INFO CAN I PUT TO A JSON DOCUMENT
TO WORK EFFICIENTLY WITH IT?

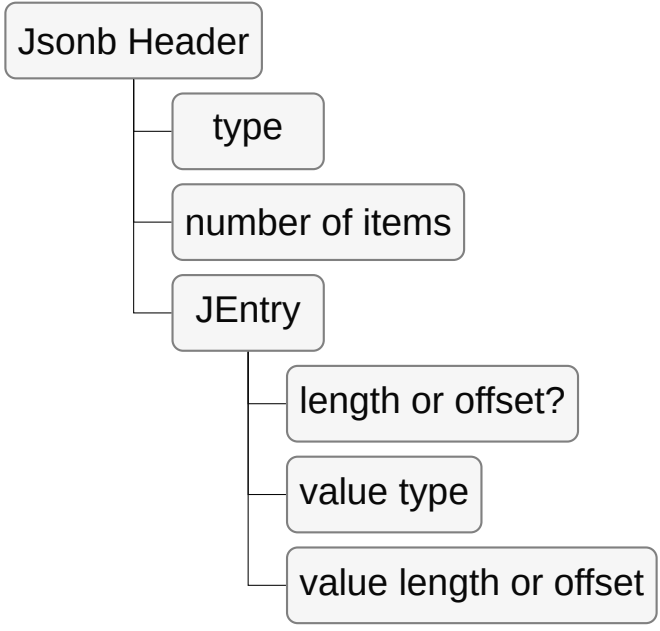
zalando



zalando



zalando


Jsonb Header
type
number of items
JEntry
length or offset?
value type
value length or offset

zalando

JB_OFFSET_STRIDE

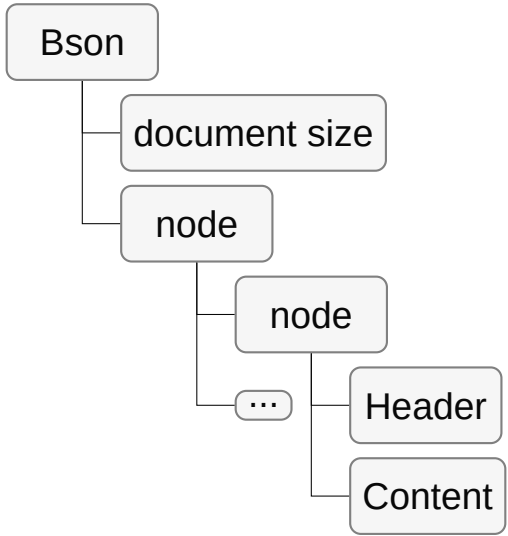
- JEntry may contains a value lenght or offset
- Offset = access speed
- Length = compressibility
- Every **JB_OFFSET_STRIDE**’th JEntry contains an offset
- Rest of them contain length

```
{"a": 3, "b": "xyz"}
```

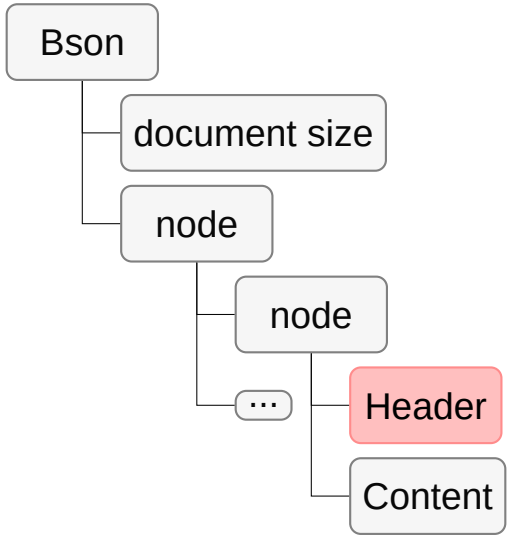
```
select pg_relation_filepath(oid),
relpages from pg_class
where relname = 'table_name';

 pg_relation_filepath | relpages
----------------------+----------
 base/40960/325477    |        0
(1 row)
```

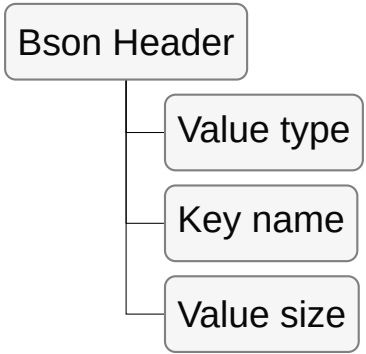
```
\x10\x03\x00\x00\x00ab\x00\x00\x00\x00\x00\x00\x80\x03\x00xyz\x00\x00\x00\x00
```



zalando



zalando

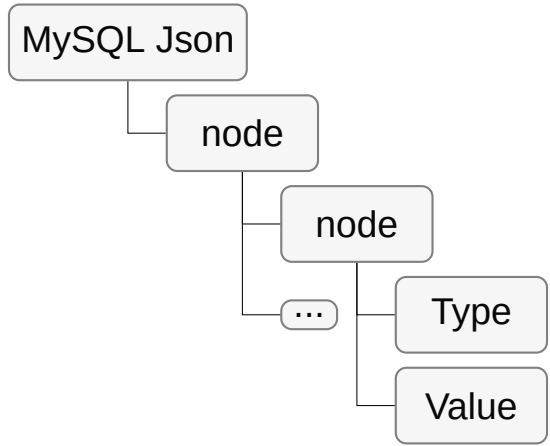


zalando

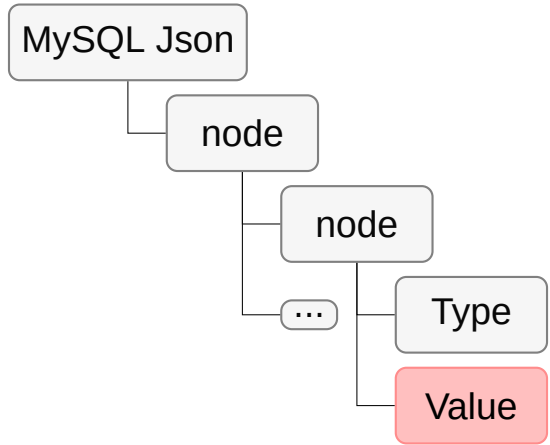
```
bson.dumps({"a": 3, "b": u"xyz"})
```

```
\x17\x00\x00\x00\x10a\x00\x03\x00\x00\x00\x02b\x00\x04\x00\x00\x00xyz\x00\x00
```

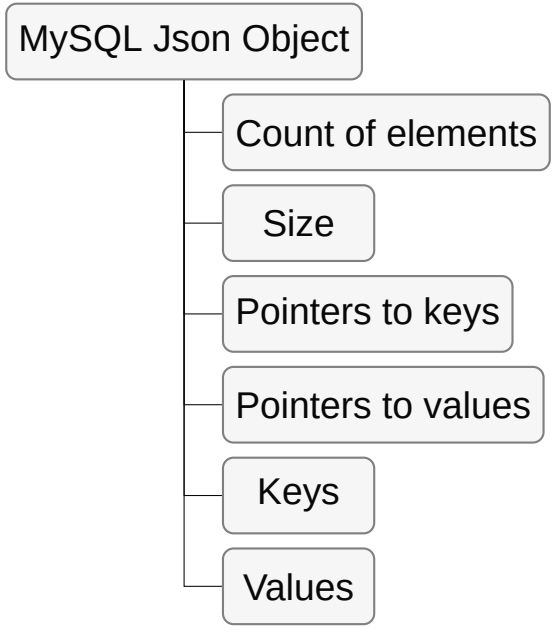

zalando



zalando



zalando

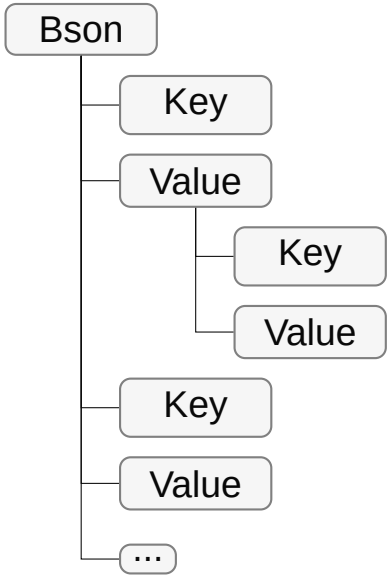

MySQL Json Object
Count of elements
Size
Pointers to keys
Pointers to values
Keys
Values

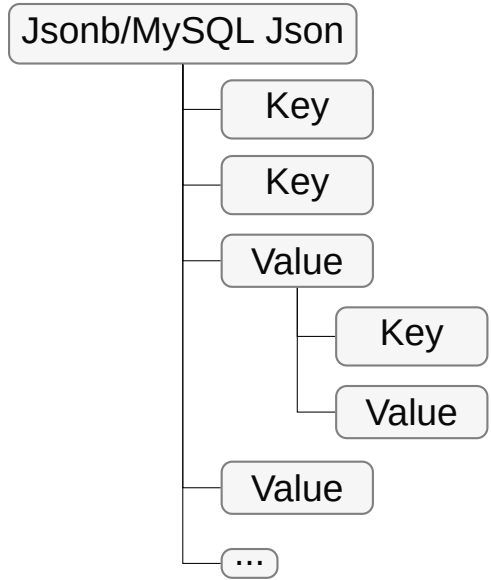
zalando

$ hexdump -C database/table.ibd

```
\x00\x02\x00\x18\x00\x12\x00\x01\x00\x13\x00\x01\x00\x05\x03\x00\x0c\x14\x00ab\x03xyz\x00
```

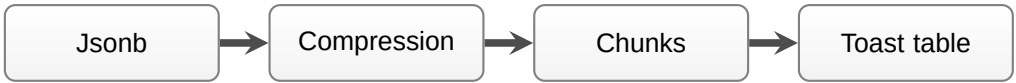

zalando


Bson
Key
Value
Key
Value
Key
Value
...


Jsonb/MySQL Json
Key
Key
Value
Key
Value
Value
...

zalando

TOAST



- TOAST_TUPLE_THRESHOLD bytes (normally 2 kB)
- PostgreSQL and MySQL use LZ variation
- MongoDB uses snappy block compression

Alignment

Variable-length portion is aligned to a 4-byte

```
insert into test
values('{"a": "aa", "b": 1}');
```

```
abaa\x20\x00\x00\x00\x00\x80\x01\x00
```

```
insert into test
values('{"a": 1, "b": "aa"}');
```

```
\x00\x00ab\x00\x00\x20\x00\x00\x00\x00\x80\x01\x00aa
```

In-memory representation

- ➔ Tree-like representation (JsonbValue, Document, Json_dom)
- ➔ Little bit more expensive but more convenient to work with
- ➔ Mostly in use to modify data (except MySQL)
- ➔ Most of the read operations use on-disk representation

Indexing support

- Postgresql – single field, multiple fields, entire document
- MongoDB – single field, multiple fields
- MySQL – virtual columns, single field, multiple fields

PG indexing details

- ➔ JGIN_MAXLENGTH
- ➔ jsonb_path
- ➔ jsonb_path_ops

Benchmarks


GREAT PERFORMANCE

AWS EC2

m4.xlarge instance

separate instance (database and generator)

16GB memory, 4 core 2.3GHz

Ubuntu 16.04

Same VPC and placement group

AMI that supports HVM virtualization type

at least 4 rounds of benchmark

PostgreSQL 9.6.3

MySQL 5.7.9

MongoDB 3.4.4

YCSB 0.9

10^6 rows and operations

AWS EC2

Configuration

shared_buffers

effective_cache_size

max_wal_size

innodb_buffer_pool_size

write concern level (journaled or transaction_sync)

Document types

“simple” document
10 key/value pairs (100 characters)

“large” document
100 key/value pairs (200 characters)

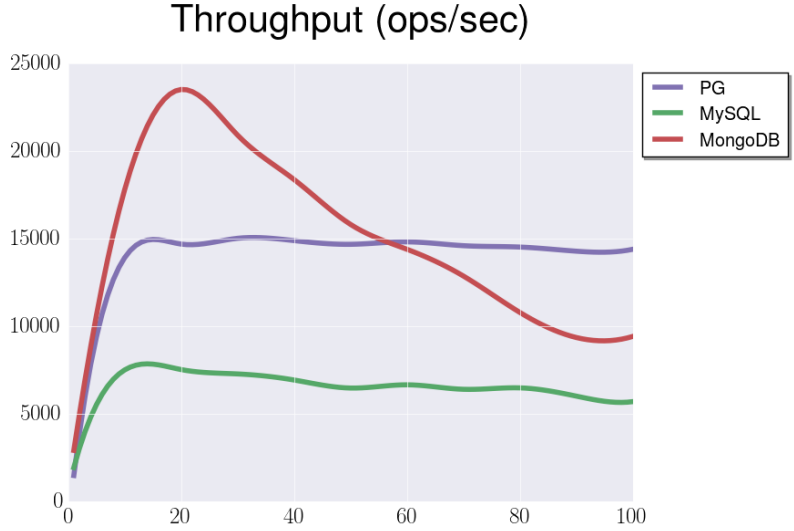
“complex” document
100 keys, 3 nesting levels (100 characters)

Select, GIN

”simple” document

jsonb_path_ops

where data @> ’{”key”: ”value”}’::jsonb


Throughput (ops/sec)
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
PG
MySQL
MongoDB

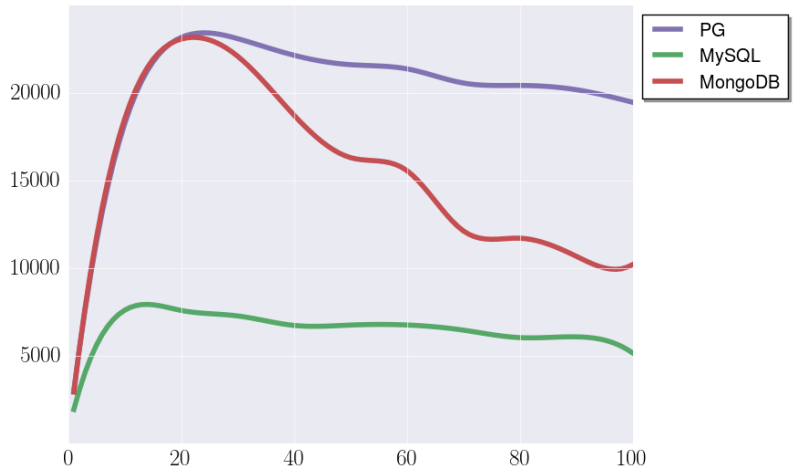
zalando

zalando

Select, BTree

”simple” document

btree

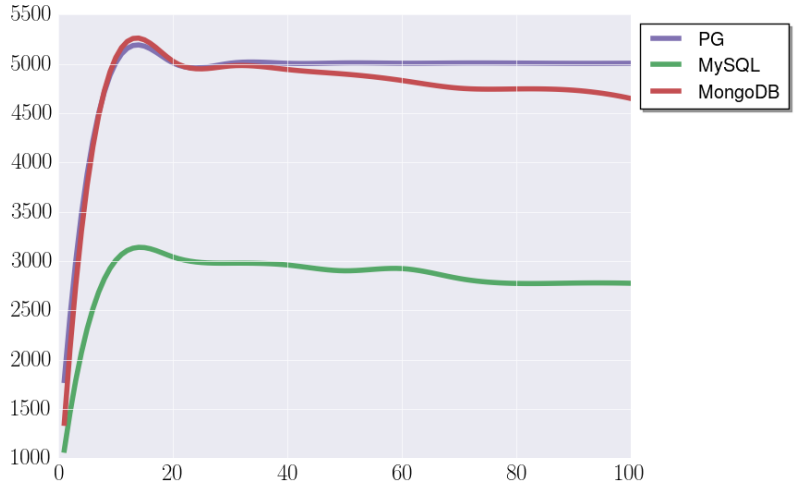

Throughput (ops/sec)
PG
MySQL
MongoDB
20000
15000
10000
5000
0
20
40
60
80
100

zalando

Select, BTree

”complex” document

btree


Throughput (ops/sec)
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
0
20
40
60
80
100
PG
MySQL
MongoDB

zalando

Scalability

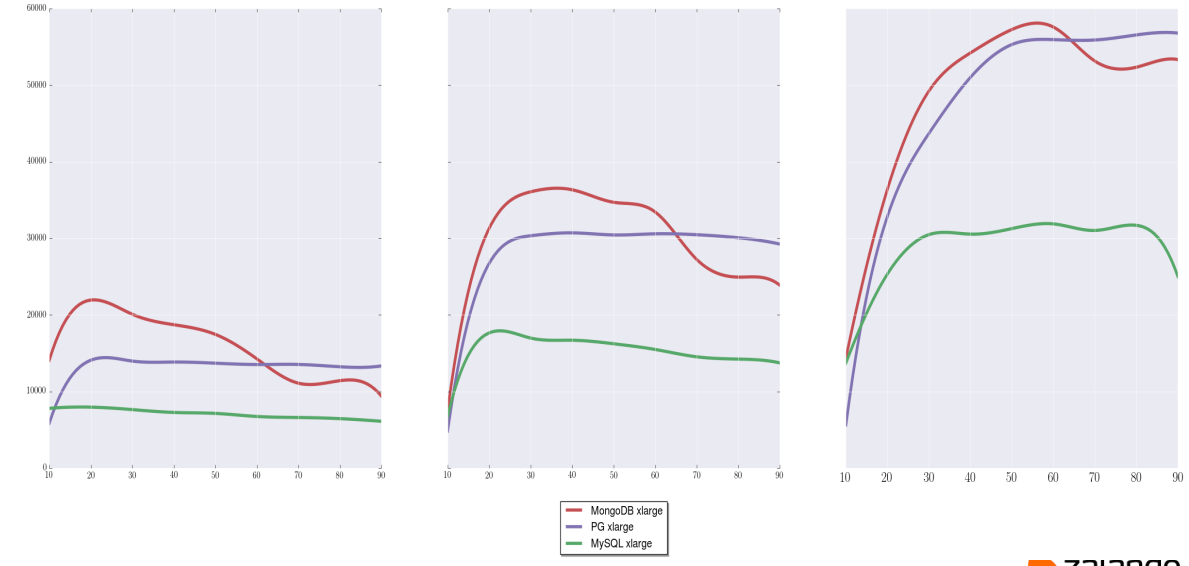
”simple” document

m4.large

m4.xlarge

m4.2xlarge

Throughput (ops/sec)

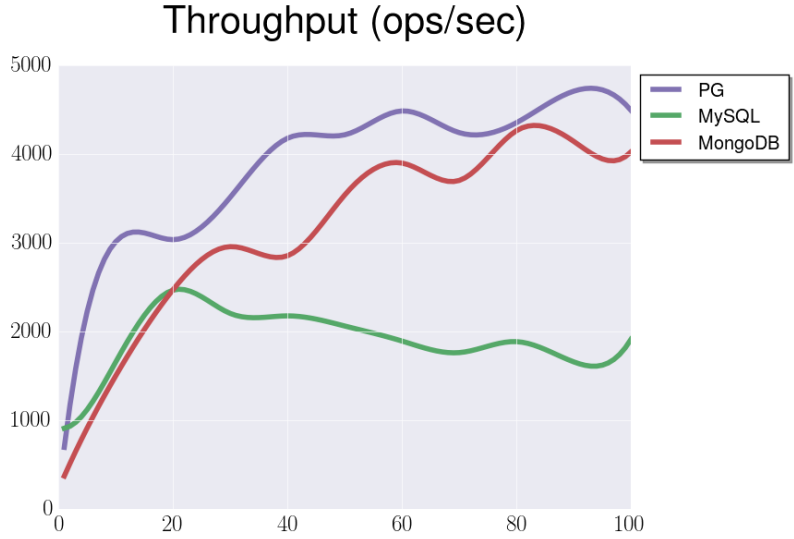


zalando

Insert

”simple” document

journaled


Throughput (ops/sec)
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
PG
MySQL
MongoDB

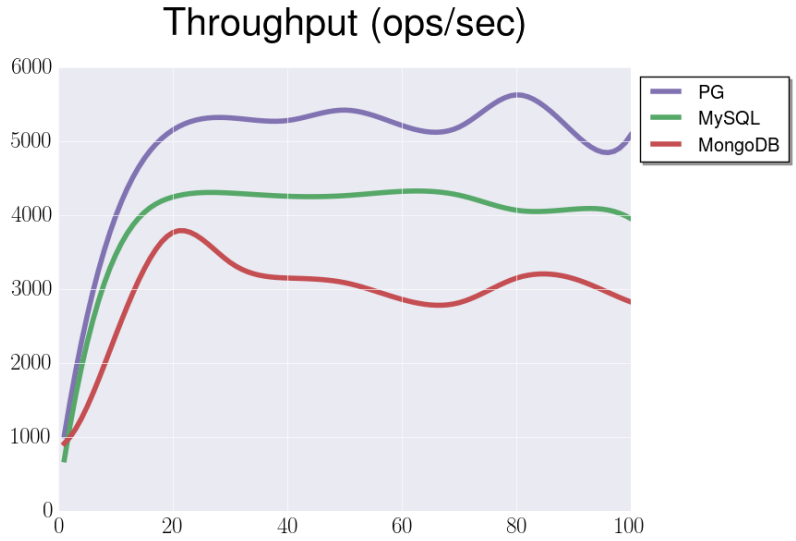
zalando

Update 50%, Select 50%

”simple” document

Update one field

transaction_sync


Throughput (ops/sec)
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
PG
MySQL
MongoDB

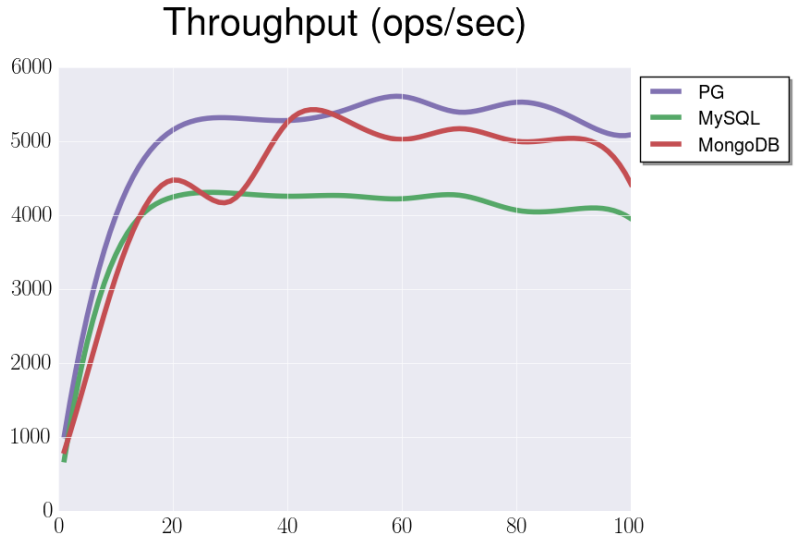
zalando

Update 50%, Select 50%

”simple” document

Update one field

journaled

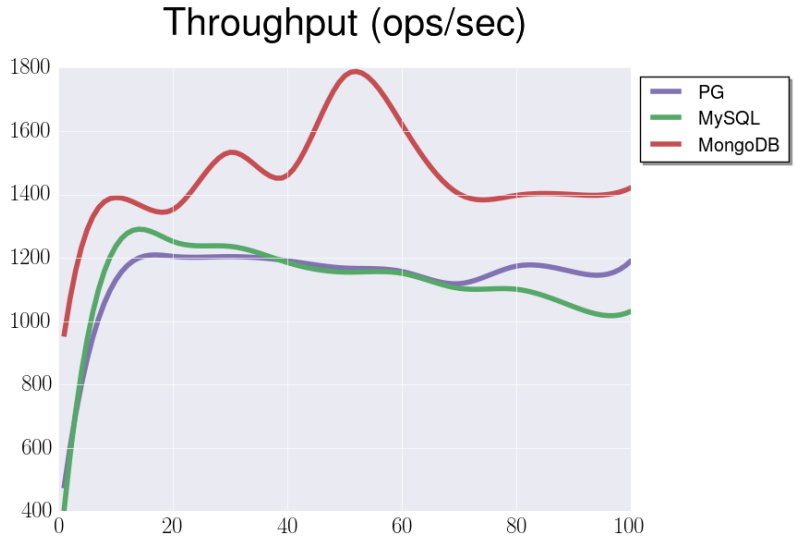

Throughput (ops/sec)
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
PG
MySQL
MongoDB

zalando

Update 50%, Select 50%

”large” document

Update one field


Throughput (ops/sec)
1800
1600
1400
1200
1000
800
600
400
0
20
40
60
80
100
PG
MySQL
MongoDB

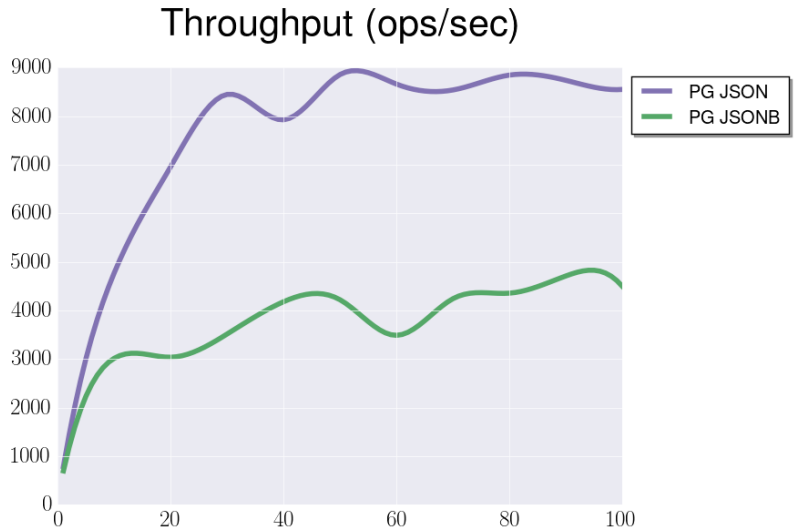
zalando

JSON vs JSONB

”simple” document

btree

insert


Throughput (ops/sec)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
PG JSON
PG JSONB

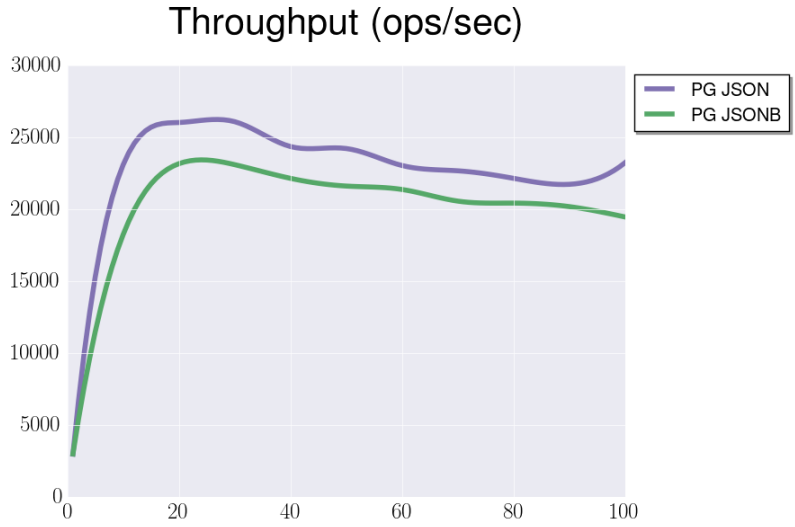
zalando

JSON vs JSONB

”simple” document

btree

select


Throughput (ops/sec)
30000
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
PG JSON
PG JSONB

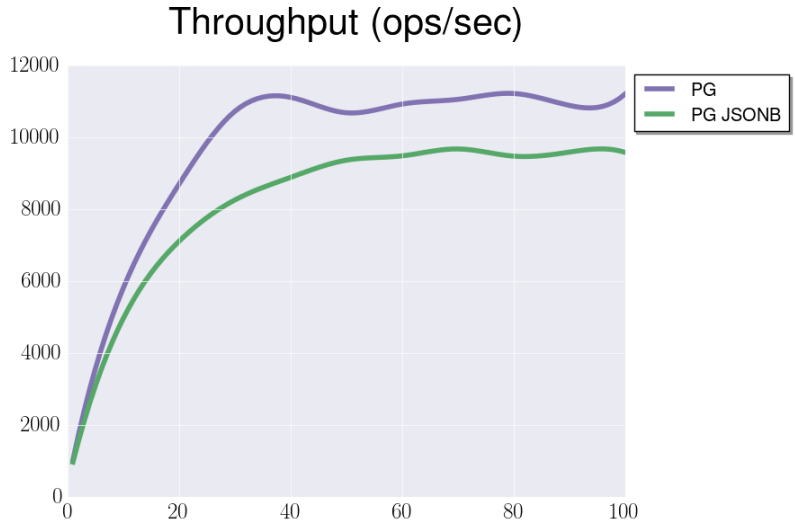
zalando

SQL vs JSONB

”simple” document

btree

insert


Throughput (ops/sec)
12000
10000
8000
6000
4000
2000
0
0
20
40
60
80
100
PG
PG JSONB

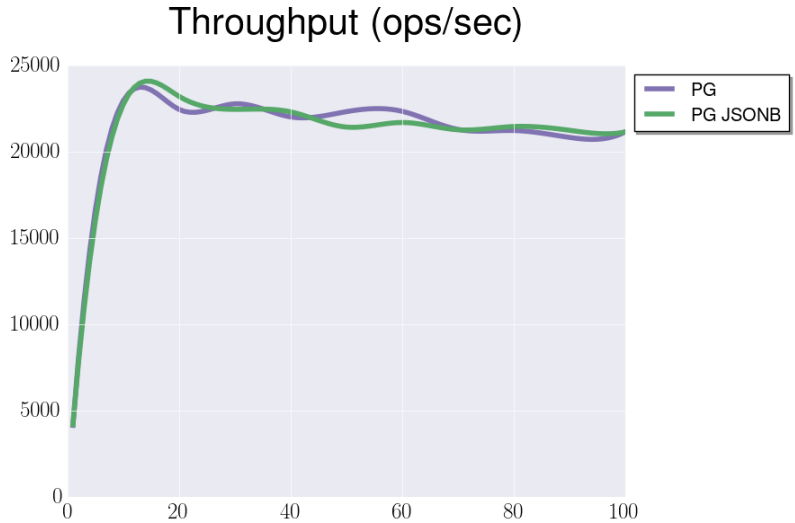
zalando

SQL vs JSONB

”simple” document

btree

select


Throughput (ops/sec)
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
PG
PG JSONB

zalando

How to bring it down acci-
dentally?

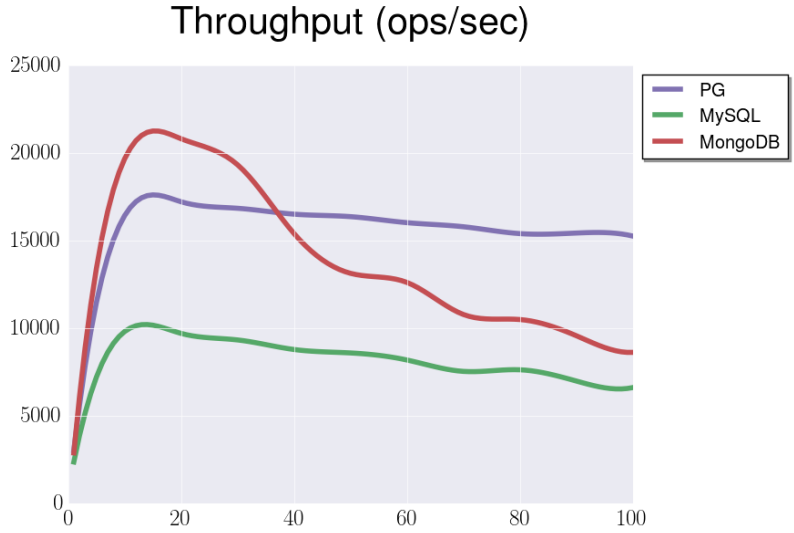
zalando

- ➔ Update one field of a document
- ➔ DETOAST of a document
  (select, constraints, procedures etc.)
- ➔ Reindex of an entire document

Document slice

”large” document

One field from a document

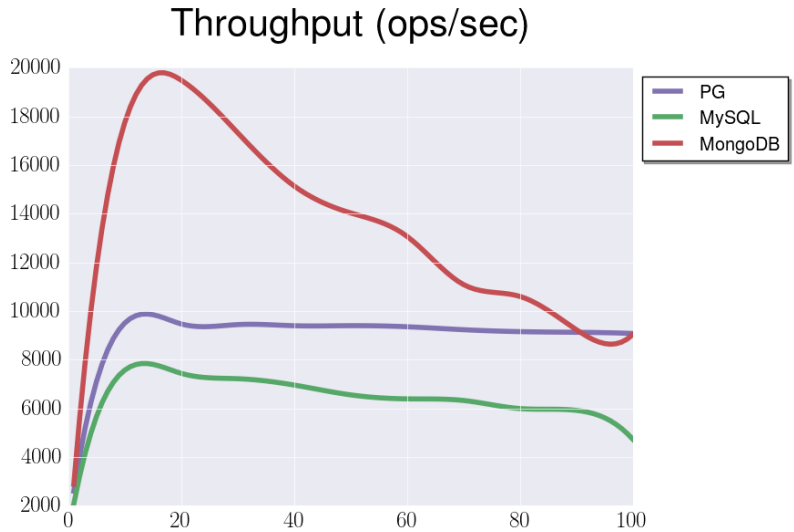

Throughput (ops/sec)
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
PG
MySQL
MongoDB

zalando

Document slice

”large” document

10 fields from a document


Throughput (ops/sec)
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
20
40
60
80
100
PG
MySQL
MongoDB

zalando

Document slice

```
create type test as ("a" text, "b" text);
insert into test_jsonb
values('{"a": 1, "b": 2, "c": 3}');
select q.* from test_jsonb,
jsonb_populate_record(NULL::test, data) as q;
 a | b
---+---
 1 | 2
(1 row)
```


SET STORAGE EXTERNAL

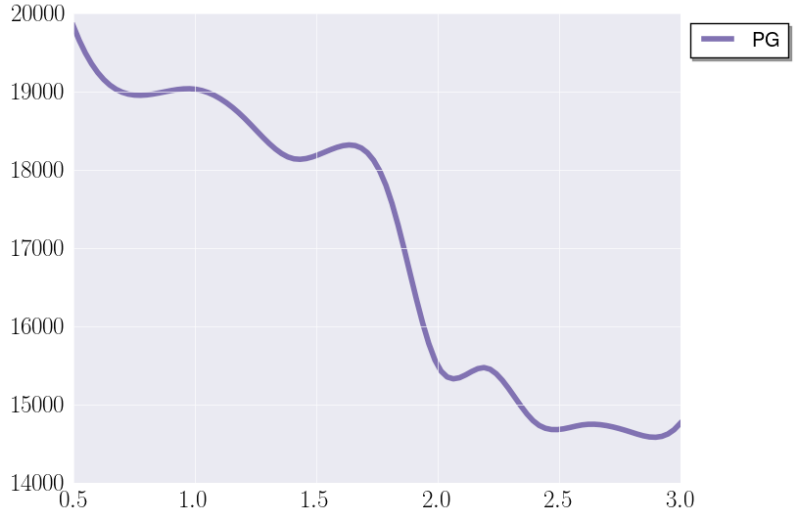
TOAST_TUPLE_THRESHOLD

”simple” document

40 threads

different document size

select


Throughput, 40 clients
20000
19000
18000
17000
16000
15000
14000
0.5
1.0
1.5
2.0
2.5
3.0
PG

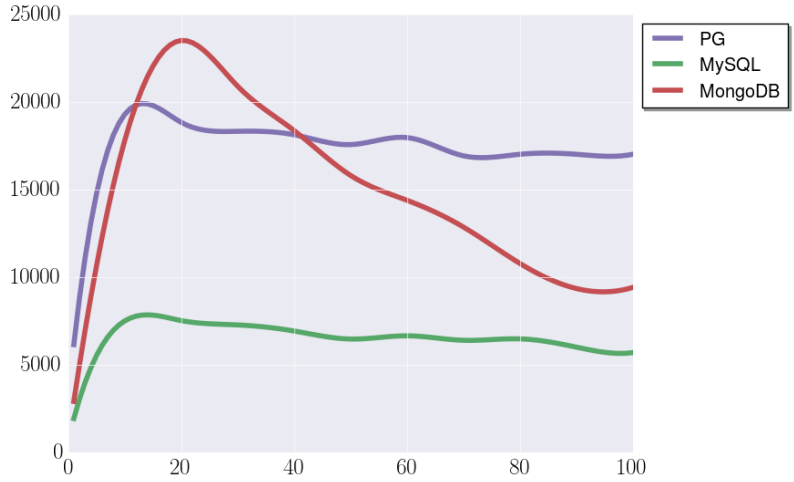
zalando

Select, GIN

”simple” document

jsonb_path_ops

where data @> jsonb_build_object(’key’, ’value’)


Throughput (ops/sec)
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
PG
MySQL
MongoDB

zalando

➔ Jsonb is more that good for many use cases

- ➔ Jsonb is more that good for many use cases
- ➔ Benchmarks above are only ”hints”

- ➔ Jsonb is more that good for many use cases
- ➔ Benchmarks above are only ”hints”
- ➔ You need your own tests

Questions?

 github.com/erthalion

 @erthalion

 9erthalion6 at gmail dot com

zalando