

Stored Procedures

The Good, the Bad and the Elegant

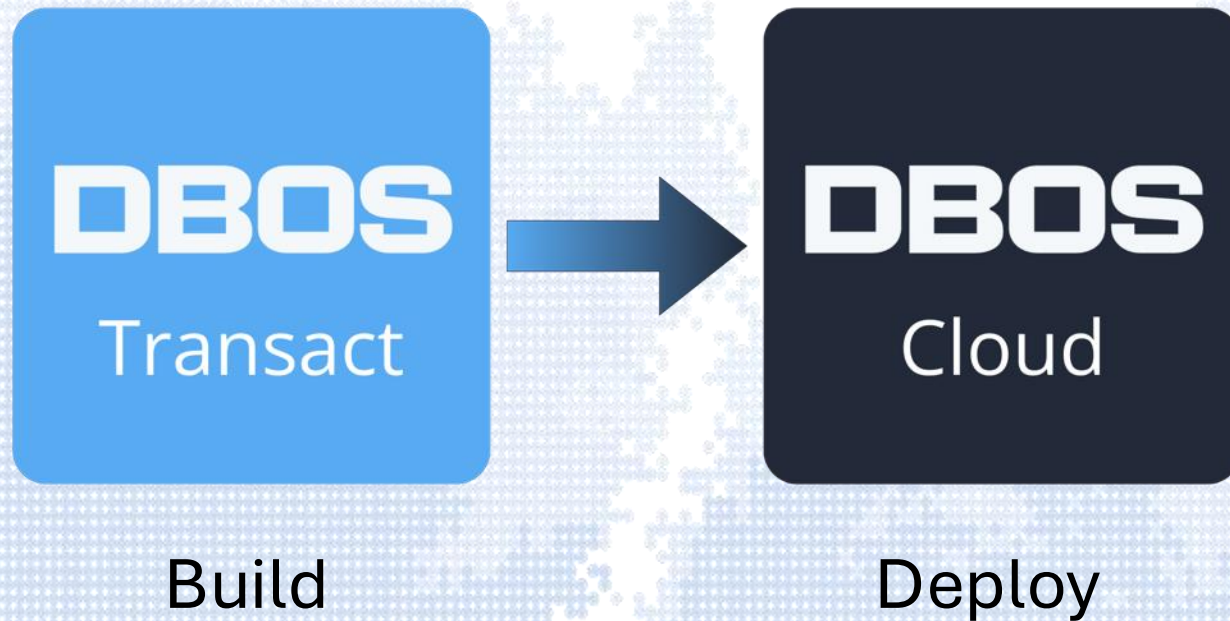
Harry Pierson, Staff Engineer
harry.pierson@dbos.dev
@DevHawk

DBOS[®]

Agenda

- Welcome
- Stored Procedures
 - The Good
 - The Bad
 - The Elegant
- Demonstration
- Q&A

About DBOS



- Open source framework
 - Easier backend development
 - Reliable, Observable, Performant
 - 10x less code
 - TypeScript, Python (soon)
- Stateful serverless platform
 - 15x more cost-efficient than AWS Lambda & Step Functions
- Use it for free!

<personal-anecdote />

**Everybody knows the benefits
of using Stored Procedures.**

*Performance and
Scalability*

3 BEGIN TRANSACTION;

6 < OA00: check for previous execution >

1 INSERT INTO orders
(last_update_time, order_status, username)
VALUES (\$1, \$2, \$3) RETURNING order_id;

2 INSERT INTO order_items
(order_id, price, product_id, quantity)
VALUES (\$1, \$2, \$3, \$4), (\$5, \$6, \$7, \$8), ...;

5 < OA00: save function result >

4 COMMIT TRANSACTION;

*Once And Only Once
(OA00)*

“Physical separation causes high communication overhead: in our experiments with OpenWhisk, communication accounts for as much as 98% of function runtime.”

“Apiary: A DBMS-Integrated Transactional Function-as-a-Service Framework”

<https://arxiv.org/abs/2208.13068>

**Everybody knows the benefits
of using Stored Procedures.**

Almost
^

**Nobody thinks those benefits
are worth it.**

Stored Procedure Friction Points

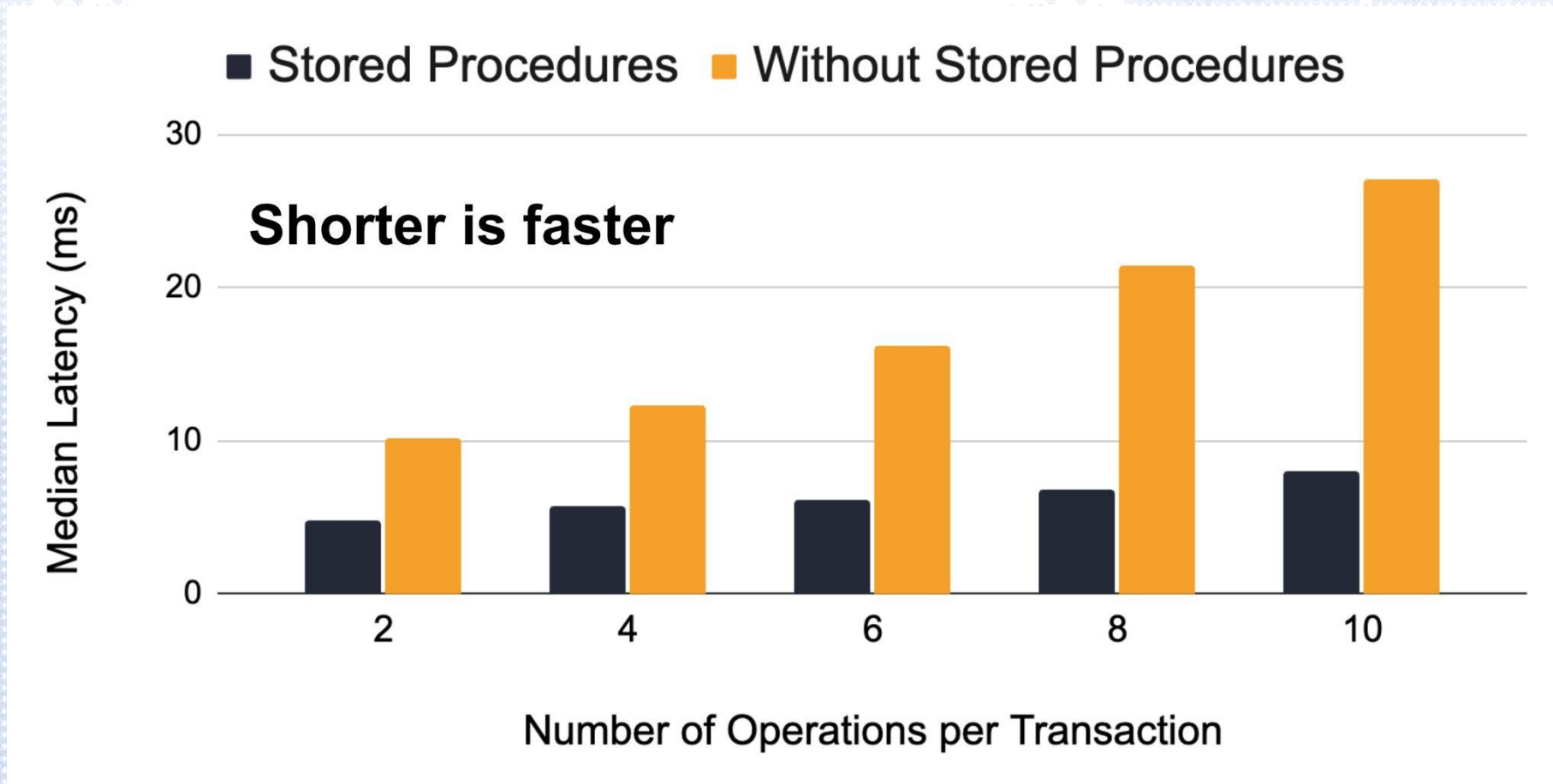
- They aren't written in same language as your application
- They don't integrate into your existing developer workflow
- They aren't versioned the same way as your application
- They are hard to debug

*These don't seem like
show-stoppers...*

DBOS @StoredProcedure

All the performance ... None of the friction

All the Performance...



...None of the Friction

- ~~They aren't written in same language as your application~~
 - DBOS @StoredProcedures are written in TypeScript
- ~~They don't integrate into your existing developer workflow~~
 - DBOS @StoredProcedures are written inline with your other DBOS application logic
- ~~They aren't versioned the same way as your application is~~
 - Deployed DBOS @StoredProcedures can be version stamped
- ~~They are hard to debug~~
 - A DBOS @StoredProcedure can be executed locally during debugging

```
@Transaction()
static async createOrder(
  ctx: TransactionContext<Knex>, username: string, products: Product[]
): Promise<number> {
  const orders = await ctx.client<Order>('orders')
    .insert({ username, order_status: PENDING, last_update_time: 0n })
    .returning('order_id');

  const order_id = orders[0].order_id;
  const items = products.map(p => ({ order_id,
    product_id: p.product_id, price: p.price, quantity: p.inventory}));
  await ctx.client<OrderItem>('order_items').insert(items);

  return order_id;
}
```

```
@StoredProcedure()
```

```
static async createOrder(  
  ctx: StoredProcedureContext, username: string, products: Product[]  
): Promise<number> {
```

```
  const query = "INSERT INTO orders (last_update_time, order_status, username)  
    VALUES ($1, $2, $3) RETURNING order_id";
```

```
  const orders = await ctx.query<Pick<Order, 'order_id'>>(query,  
    [0n, PENDING, username]);
```

```
  const order_id = orders.rows[0].order_id;
```

```
  for (const p of products) {
```

```
    const query = "INSERT INTO order_items (order_id, product_id, price, quantity)  
      VALUES ($1, $2, $3, $4)";
```

```
    await ctx.query(query, [order_id, p.product_id, p.price, p.inventory]);
```

```
  }
```

```
  return order_id;
```

```
}
```

DBOS Compiler, Under the Hood

- Extracts @StoredProcedure methods
 - Eliminates DBOS methods except @StoredProcedure
 - Tree shakes out code that isn't used by @StoredProcedures
 - Convert await calls to be synchronous (plv8 limitation)
 - Removes decorators
 - Compiles remaining code to JavaScript via TSC
- Generate Functions and Stored Procedures
 - Every module becomes a plv8 function
 - Every @StoredProcedure function becomes a plv8 function and a pl/pgSQL stored procedure
 - Generated functions/procedures get optionally version prefixed

@StoredProcedure Demo

Good **Stored Procedure Performance**

Bad **Stored Procedure Developer Experience**

Elegant **DBOS @StoredProcedure**

harry.pierson@dbos.dev
@DevHawk