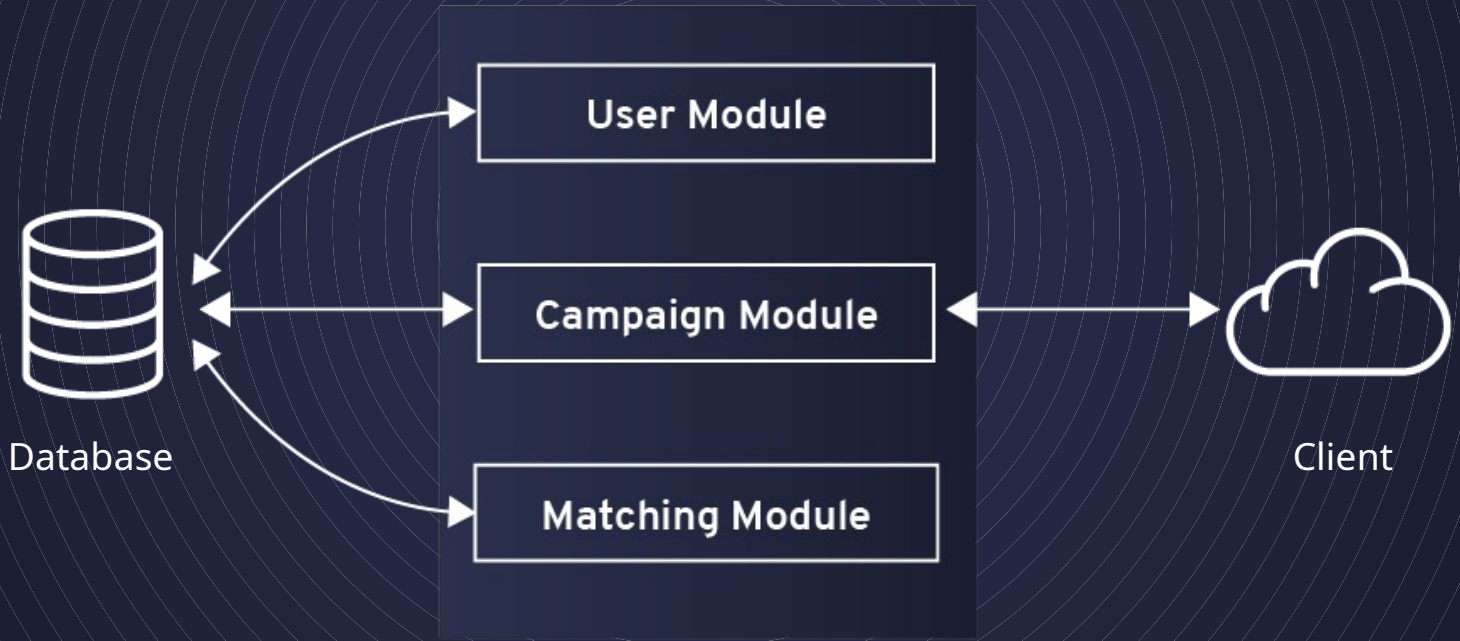
Vitess Transactional Microservices

Dan Kozlowski
PlanetScale

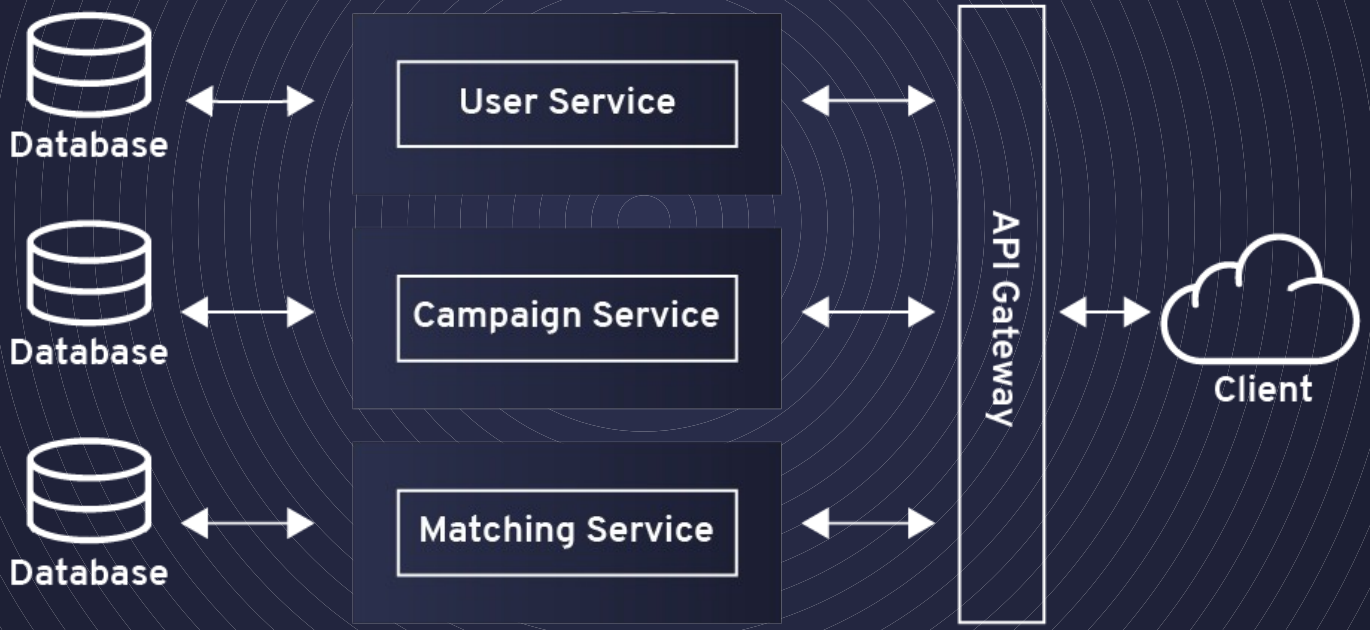
Traditional Monolithic Architecture



Vitess

Traditional Transaction Architecture

Traditional Micro-Service Architecture

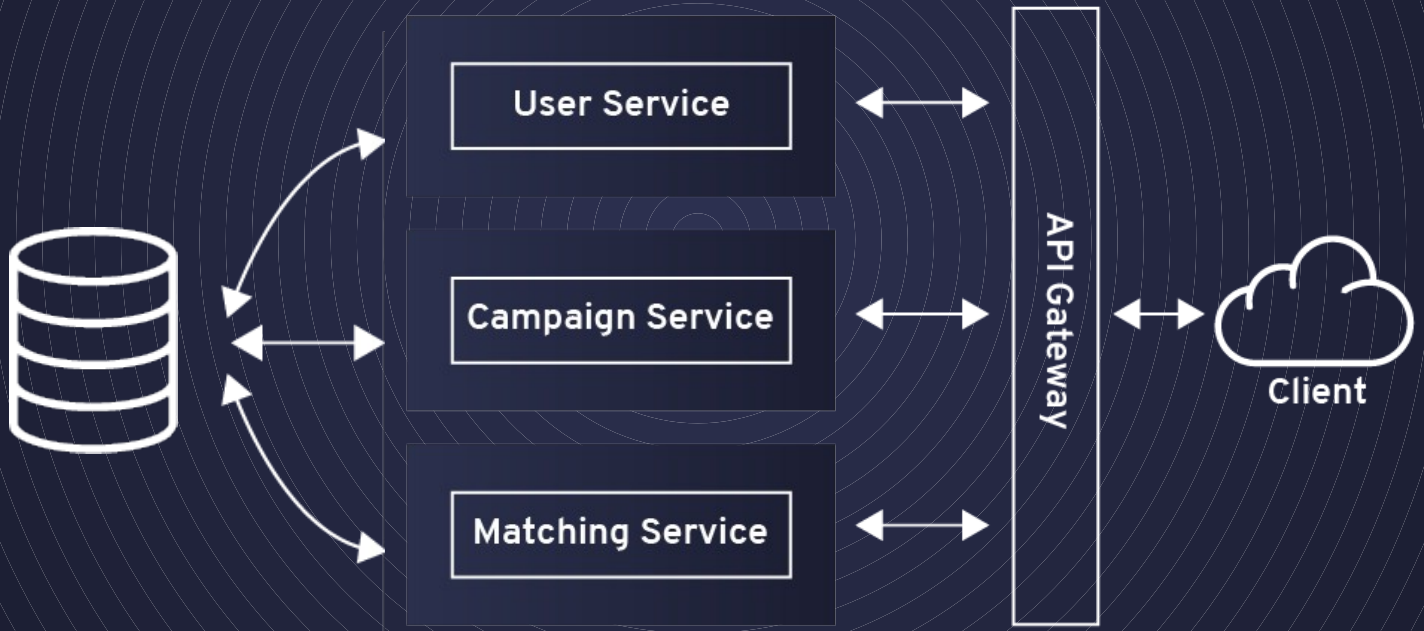


Vitess

Transactions With Microservices

Vitess

Why Not Micro-Service Architecture



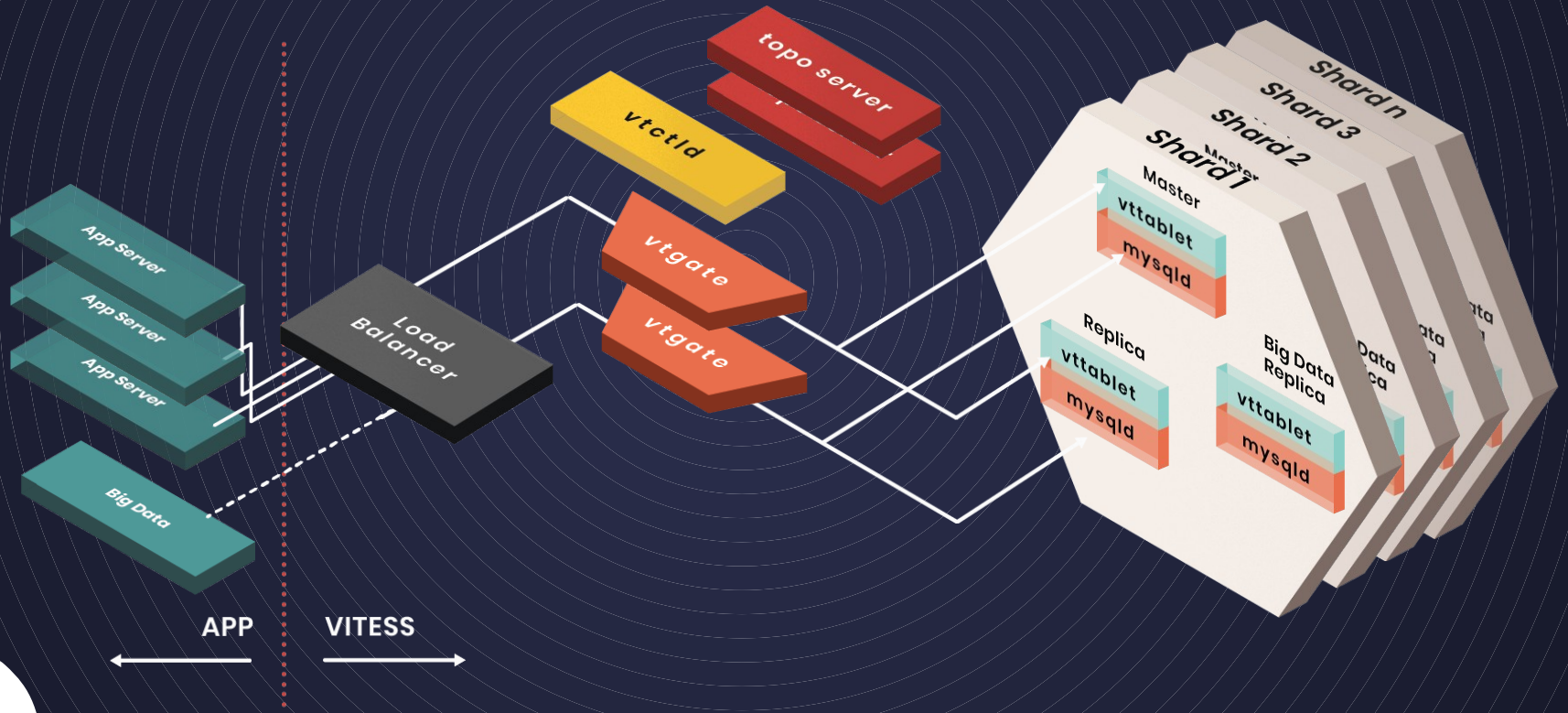
Vitess

Explicit State Management

1. Database Connection
2. TCP Connection to Client

Enter Vitess

Architecture



Vitess

gRPC Endpoint

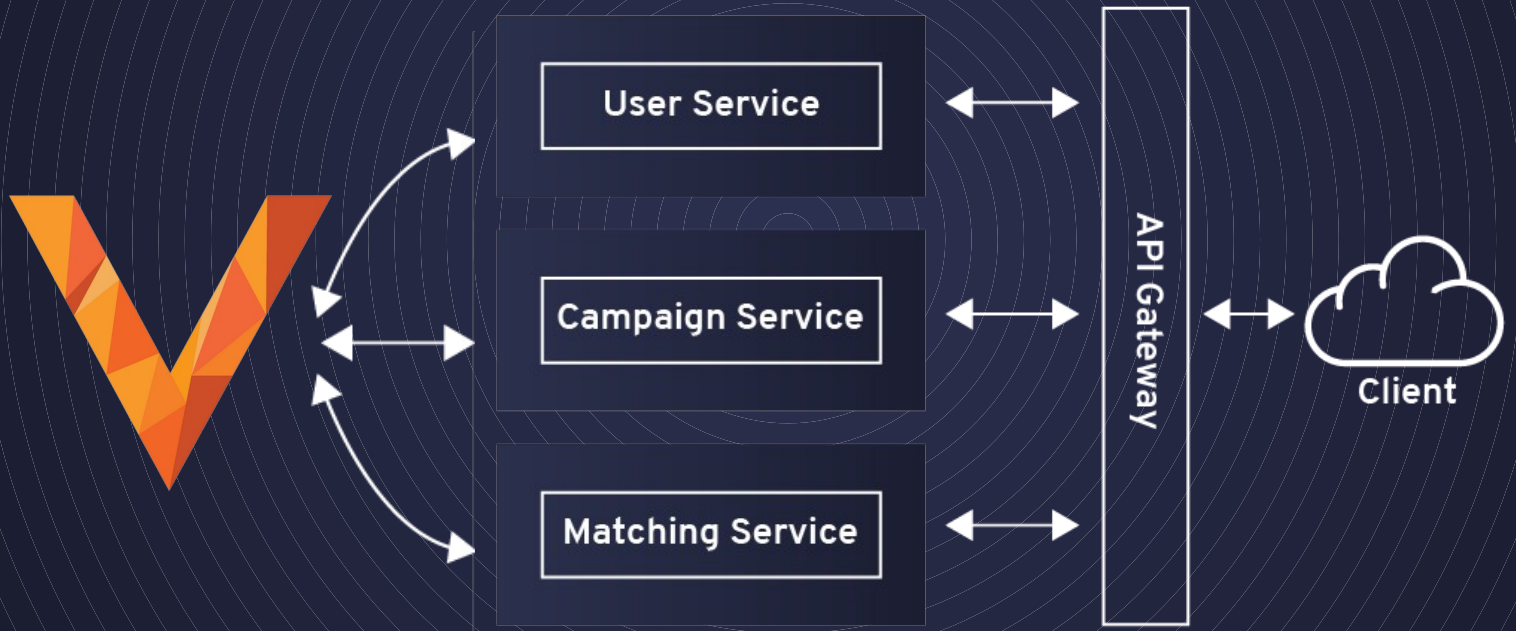
```
// ExecuteRequest is the payload to Execute.
message ExecuteRequest {
  vtrpc.CallerID caller_id = 1;
  Session session = 2;
  query.BoundQuery query = 3;
}

// ExecuteResponse is the returned value from Execute.
message ExecuteResponse {
  vtrpc.RPCError error = 1;
  Session session = 2;
  query.QueryResult result = 3;
}
```

gRPC Endpoint

```
// Session objects are exchanged like cookies through various
// calls to VTGate. The behavior differs between V2 & V3 APIs.
// V3 APIs are Execute, ExecuteBatch and StreamExecute. All
// other APIs are V2. For the V3 APIs, the session
// must be sent with every call to Execute or ExecuteBatch.
// For the V2 APIs, Begin does not accept a session. It instead
// returns a brand new one with in_transaction set to true.
// After a call to Commit or Rollback, the session can be
// discarded. If you're not in a transaction, Session is
// an optional parameter for the V2 APIs.
message Session {
  …
}
```

Vitess Micro-Service Architecture



But,

My Application is a Monolith and we are moving to Microservices!

But,

If you use the same database you will not have independent services!

But,

We could do the same thing with CQRS or Saga!

Daniel Kozlowski

@MinisterOfEng

PlanetScale

Maintainer, Vitess

https://vitess.io/slack

Vitess