

DBOS +  CockroachDB

Are your agents on ACID ??

Agenda & Speakers

01 50 Years of Application Architecture Evolution

02 Building reliable software you can trust

03 DBOS Architecture

04 Observability

05 How CockroachDB and DBOS unblock workflow execution

06 What makes CockroachDB special

07 Q&A



Mike Stonebraker
Creator of Postgres



Qian Li
CEO & Co-Founder, DBOS



David Joy
Technical Director - AI, Ecosystem & Channels, Cockroach Labs

DBOS + CockroachDB

50 Years of Application Architecture Evolution

Mike Stonebraker

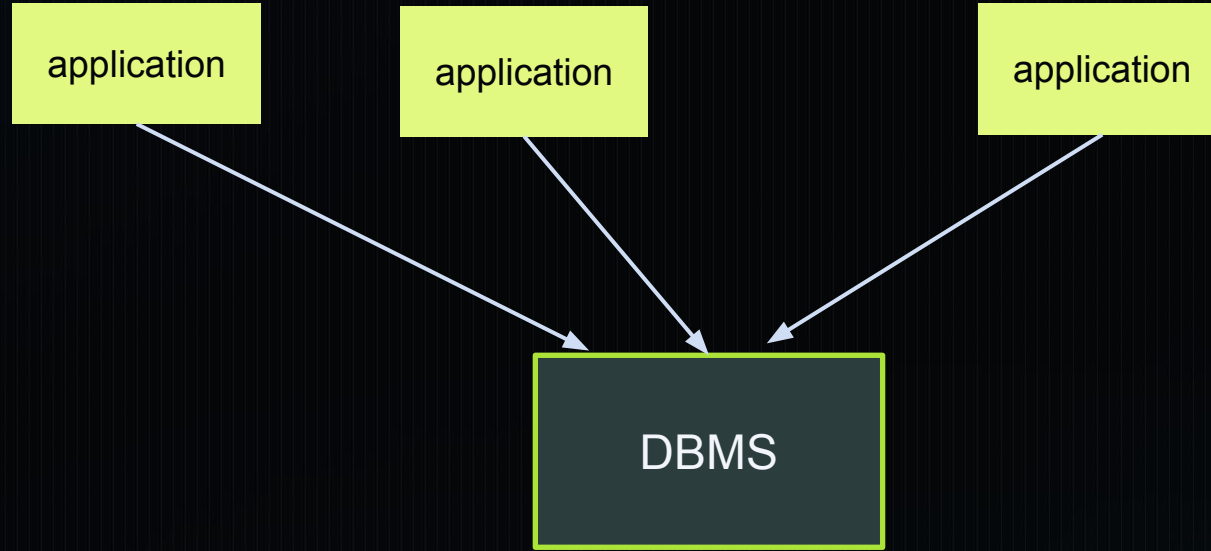
In the 1970s

- **I built Ingres**
 - With System R, showed that RDBMS was feasible
- **Designed for**
 - Business data processing
 - With the standard data types (integers, floats and character strings) hardcoded
 - Did not work well on geographic data

In the 1980s

- **I built Postgres (Post Ingres)**
- **Designed for**
 - Extensibility (geographic data, various semantics of time, etc.)
 - Time travel (our implementation sucked)
 - A rule system (for triggers, integrity constraints, referential integrity)
 - See my 2014 Turing Award lecture for a blow-by-blow account

Ingres/Postgres Architecture

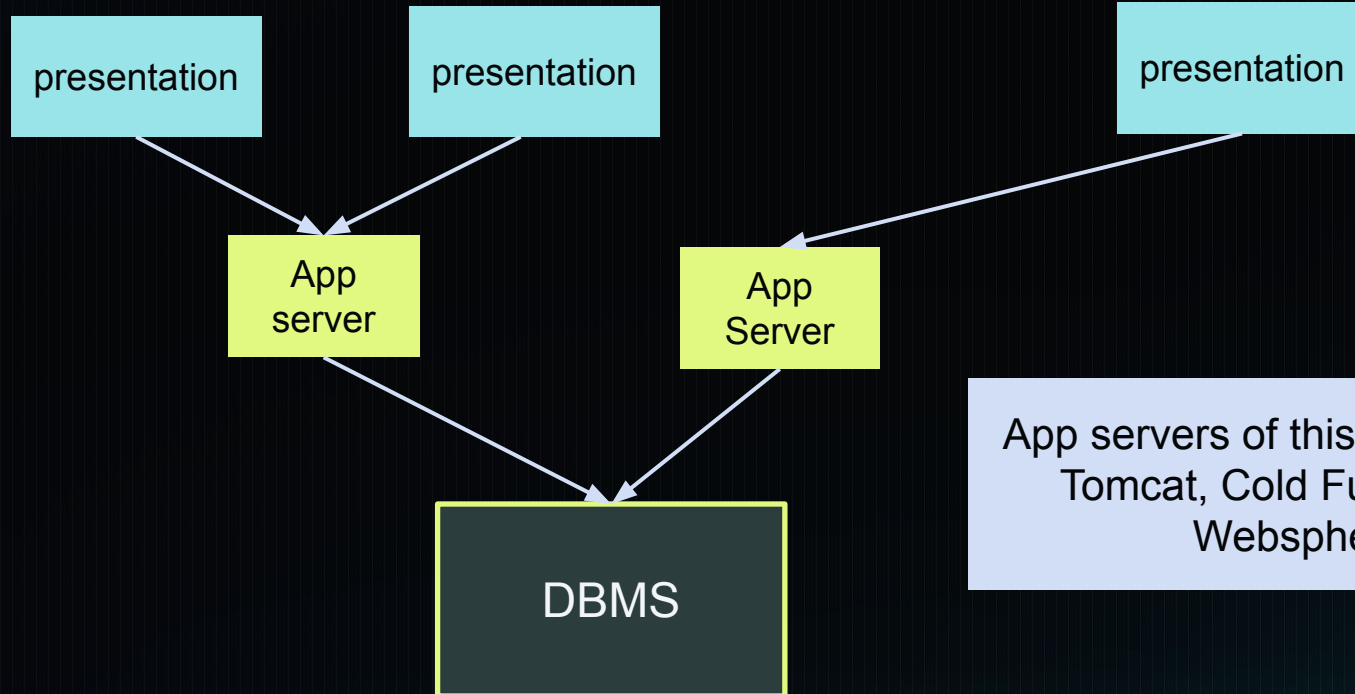


Process-per-user (preceded threads, VMs)
Postgres still works this way

Process Per User

- **Serious scalability issues as transactions-per-second increased**
 - Main memory footprint (for the application and the DBMS)
 - Not engineered for lots of connections
- **Led quickly in the 1990s to a**
 - Three level architecture

Ingres/Postgres Architecture



App servers of this era included Tomcat, Cold Fusion, and Websphere

So What Has Happened in the last 30 years?

- **Agentic AI**
 - In case you have been asleep the last couple of years
- **Heavier OLTP applications**
- **Much larger TPS demands**
 - Think Spotify, Uber, Walmart . . .

Agentic AI

- First there were LLMs (~3 years ago)
- Very quickly developers realized they needed stuff around LLMs to be successful
- So agentic AI became a **workflow** (directed graph) of “steps,” some of which were LLMs, perhaps with loops and alternate paths
- Very heavyweight (just look at your Anthropic or OpenAI token bill)
- And often you needed to backup and try something else

Agentic AI Requirements

- **Deal with failures (crashes, step problems, . . .)**
- **Bad idea**
 - Restart the workflow from the beginning
 - Expensive!!!
 - May be non-deterministic
- **Much better idea**
 - Make each step **durable**, so agentic AI can be restarted in the middle
 - No-brainer to put “state” in a DBMS – after all “durable” is the D in ACID

Agentic AI Requirements

- **Deal with step failures**
 - Need backout across transaction boundaries
 - Need **compensation** (Read pioneering paper by Garcia-Molina and Salem in SIGMOD '87)
 - See paper in CIDR '26 for a complete discussion of backout, compensation, durability, atomicity (A in ACID), and a new definition of consistency
 - ACID-2 for applications!

What About OLTP?

- **Applications are workflows of transactions**
 - Reserve inventory
 - Check credit
 - Accept payment
 - Fulfillment
- **Often at high volume**

What About OLTP?

- Concurrency control is tricky – don't want a single monster transaction
- Need backout/compensation across transaction boundaries (for example if a customer gives a bad address for fulfillment)
- Looks a lot like agentic AI – with the same crash recovery issues

In Summary

- **Need to extend app servers with features for workflow support**
 - Durability
 - Sagas
 - Atomicity
 - . . .
- **I.e. we need to move from app servers to **workflow servers****
 - lots of implementation choices and products (Airflow, DBOS, Temporal . . .)
 - In my opinion, best option is to put step state in a DBMS
 - And co-locate orchestration with steps (10x in performance)
- **Qian & David will provide more details on an implementation of this**



Build reliable software you can trust

Never lose progress

DBOS Overview

- **Key idea:** Turn your database into a durable workflow server
 - Checkpoint your program's execution state in your database so it can resume where it left off
- Use cases: AI agents, ETL pipelines, business workflows



DBOS: Effortless Reliability & Observability

```
widget_store.py

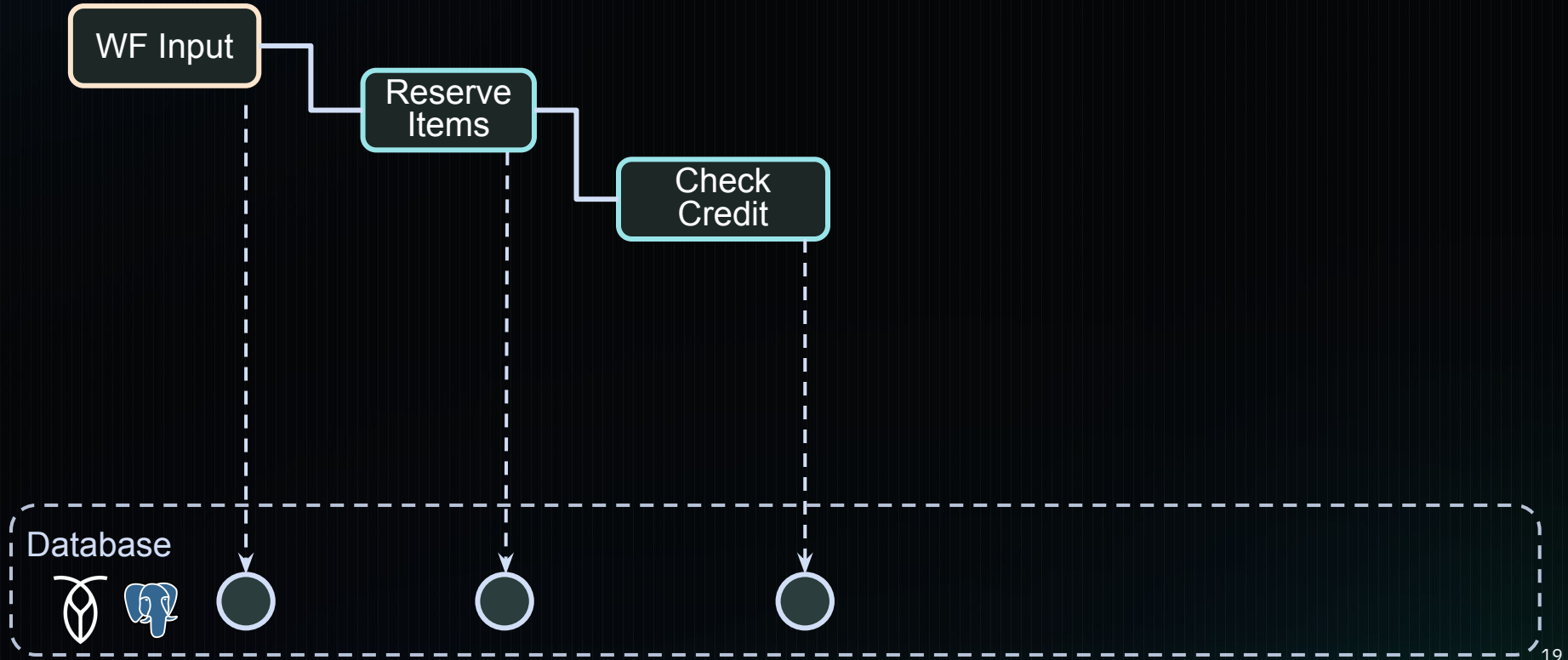
@DBOS.step()
def create_order():
    ...

@DBOS.workflow()
def checkout(items):
    order = create_order()
    reserve_inventory(order, items)
    payment_status = process_payment(order, items)
    if payment_status == 'paid':
        fulfill_order(order)
    else:
        undo_reserve_inventory(order, items)
        cancel_order(order)
```

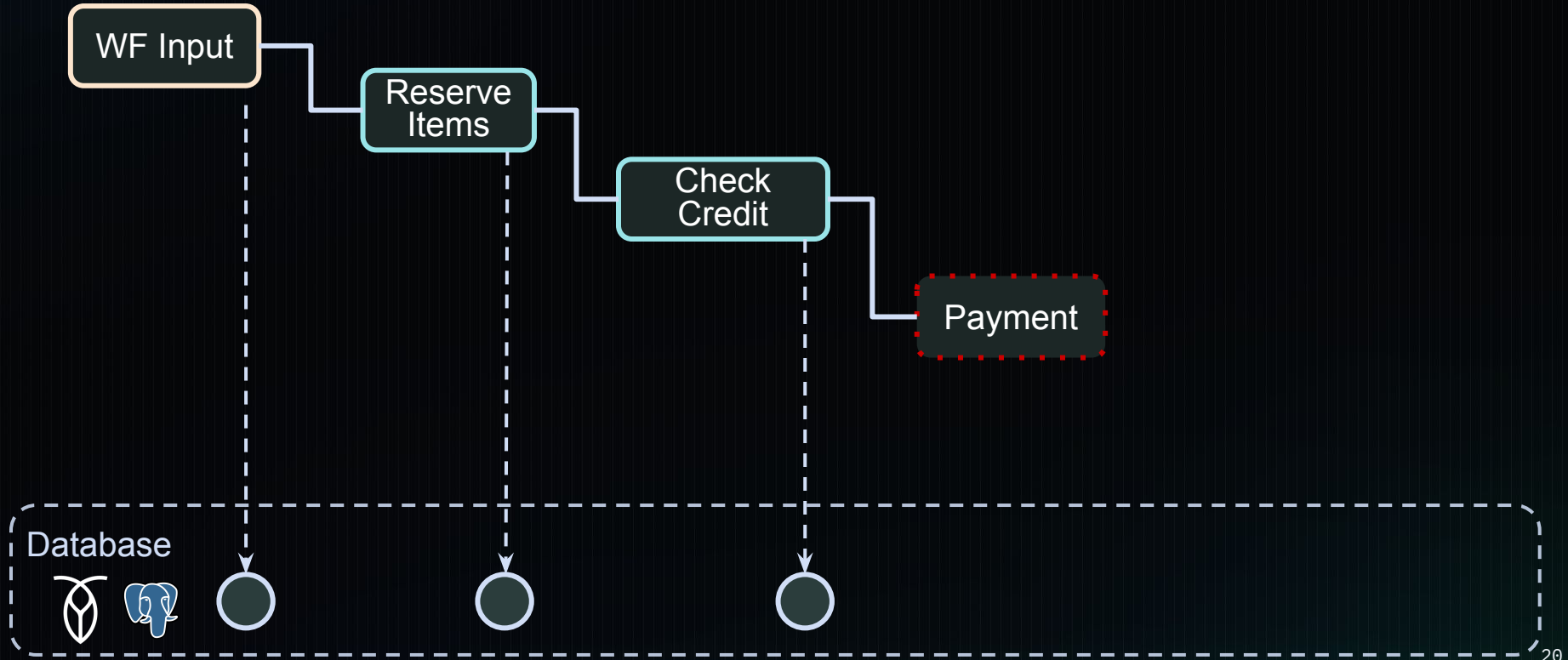
- Durable execution library for Java, Python, Go, Typescript
- Simple annotations to your code
- No need to re-architect your app
- No 3rd-party runtime dependencies

DBOS is the easiest path to reliability

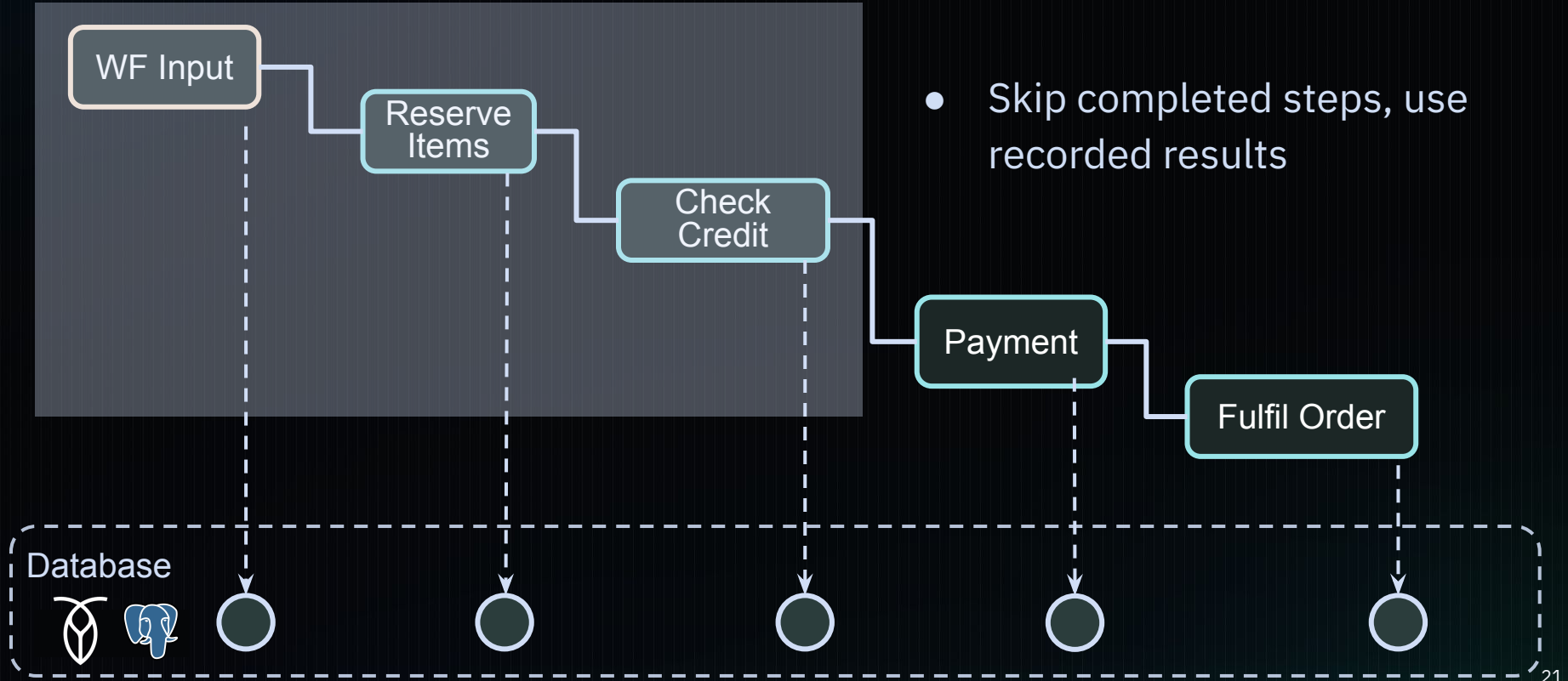
How It Works: Persist State in a Database



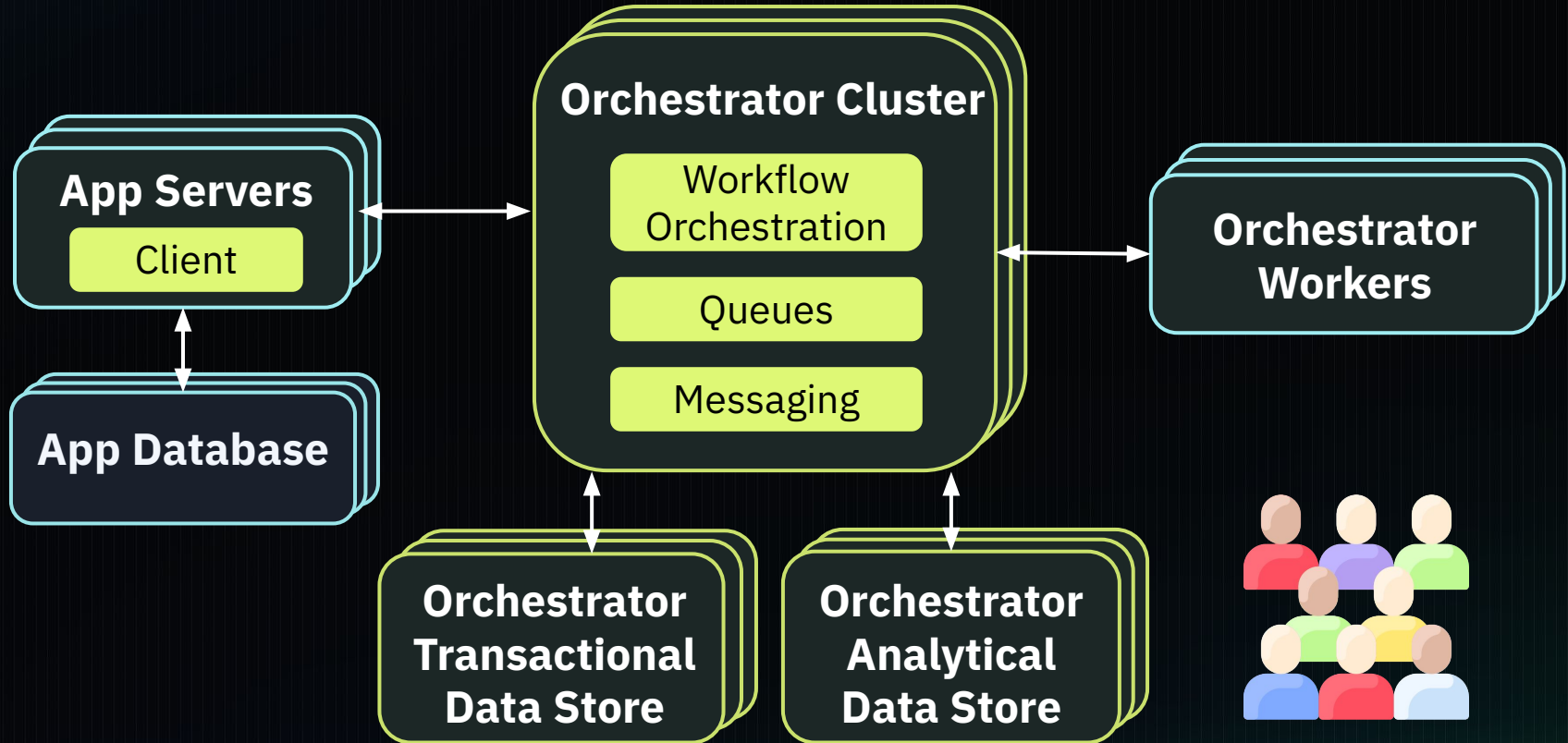
How It Works: Persist State in a Database



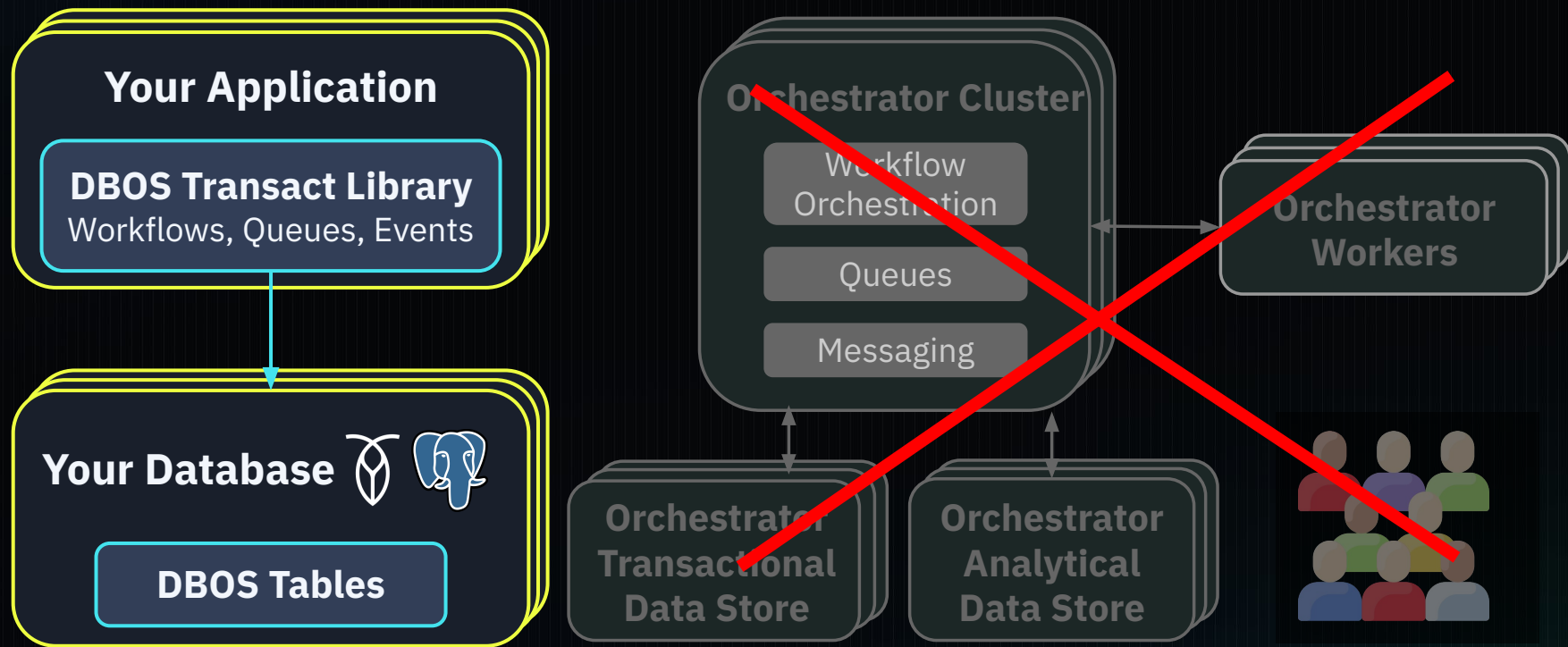
How It Works: Resume from Where Left Off



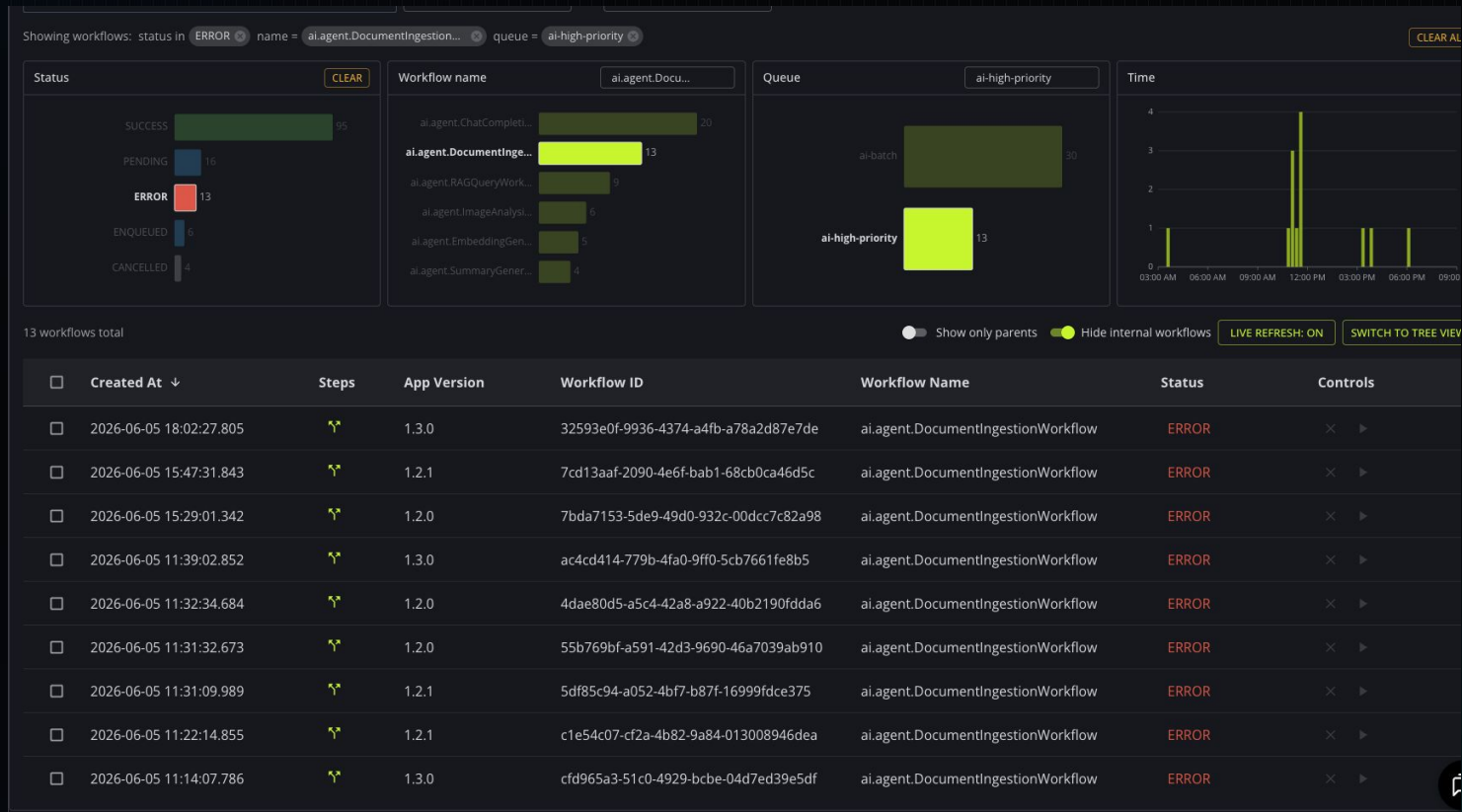
A Common Design: External Orchestrator



DBOS Architecture: Your Database Is All You Need



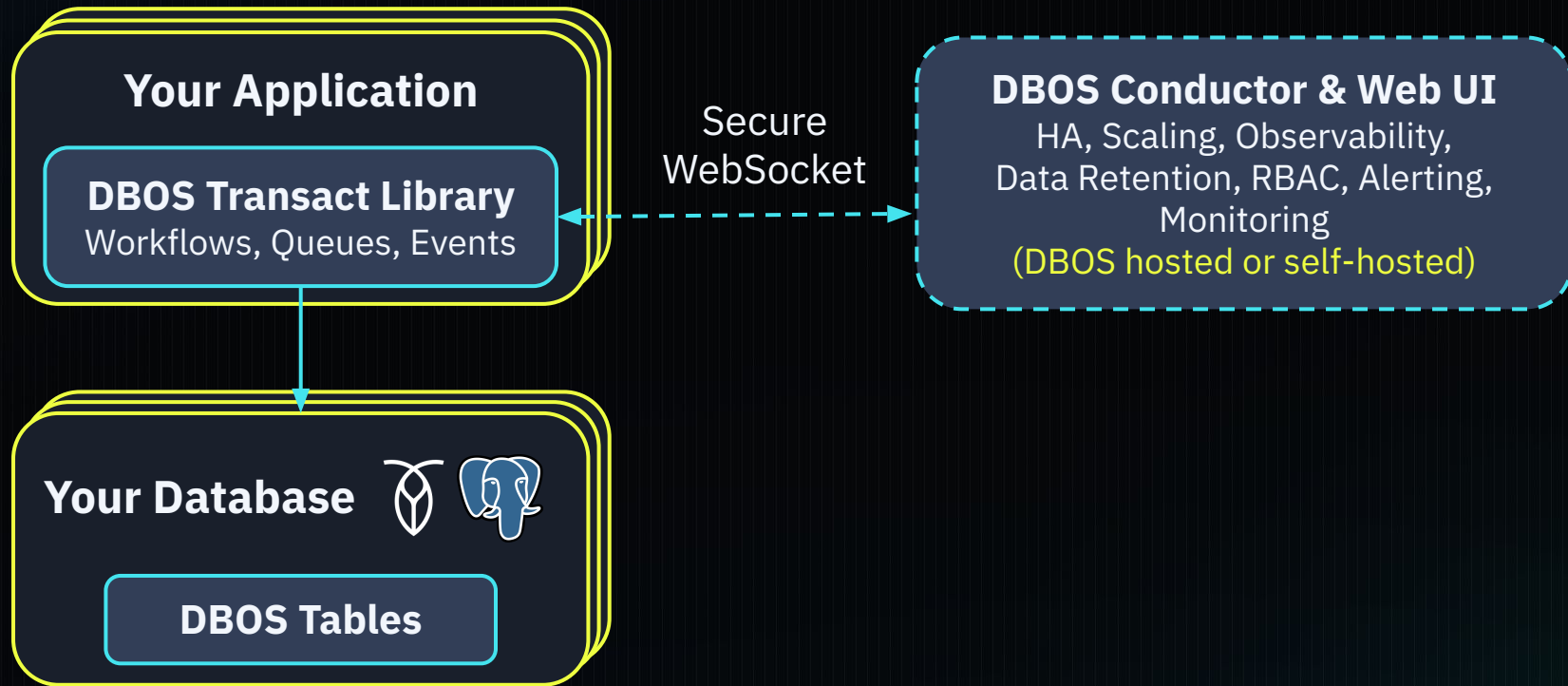
Database-Backed Observability



More Database-Backed Features

- Workflow control: fork (branching), cancel, resume, ...
- Durable queues with rate limiting, debouncing, concurrency control
- Durable sleep
- Durable events and streams
- ...

Run Confidently in Production with Conductor



Reliability At Scale with CockroachDB and DBOS

- DBOS library requires only CockroachDB (or any Postgres-compatible DB)
- Effortless AI reliability & observability at scale
- Instant compliance with regulatory audit requirements
- Save \$\$\$ and simplify infra versus dedicated orchestration servers

Infrastructure can't keep up or stay up

83%

Predict Infrastructure Failure

Within 24 months due to AI demand without major upgrades

34%

Expect Failure Sooner

Infrastructure failure anticipated within the next 11 months

98%

High Downtime Costs

One hour of AI downtime costs at least \$10,000 for nearly all firms

"Leaders aren't worried about whether AI works. They're worried their infrastructure won't hold up once AI is always on... resilience isn't about recovery — it's about whether the architecture can handle continuous success without breaking."

Why CockroachDB

ACID Compliant, SQL

Global consistency with serializable isolation. Memory is always accurate, never stale.

Always-on resilience

Survives node, zone, and region failures automatically. Agents never lose their memory.

Horizontal scale

Scales with your agent fleet without sharding or re-architecture. Prototype to production.

Native vector search

pgvector support enables semantic memory retrieval inside the same database.

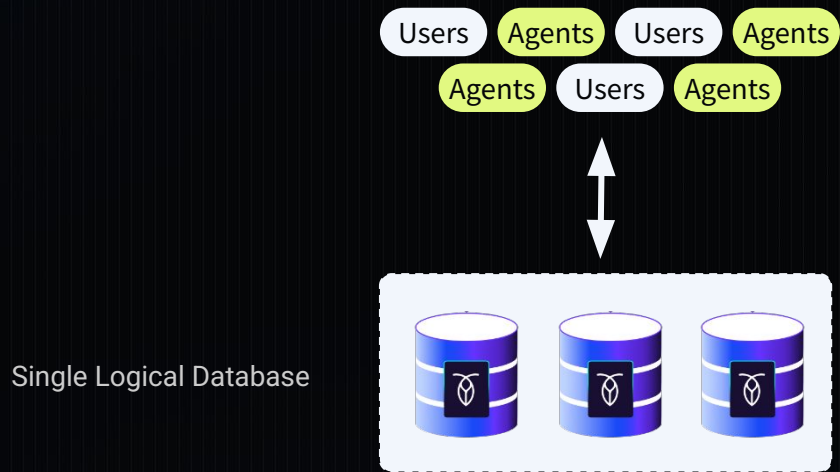
Transactional guarantees

ACID transactions ensure atomic, consistent memory writes across agents.

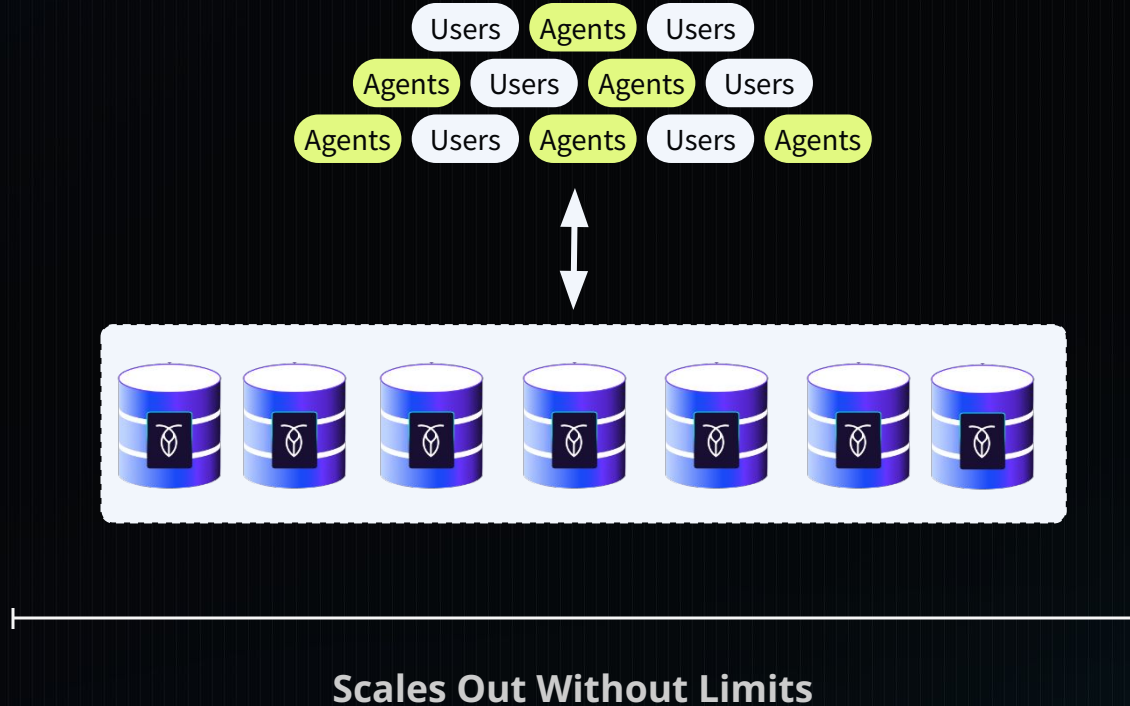
Agent-ready platform

Managed MCP Server and Agent Skills give AI agents a secure interface to data.

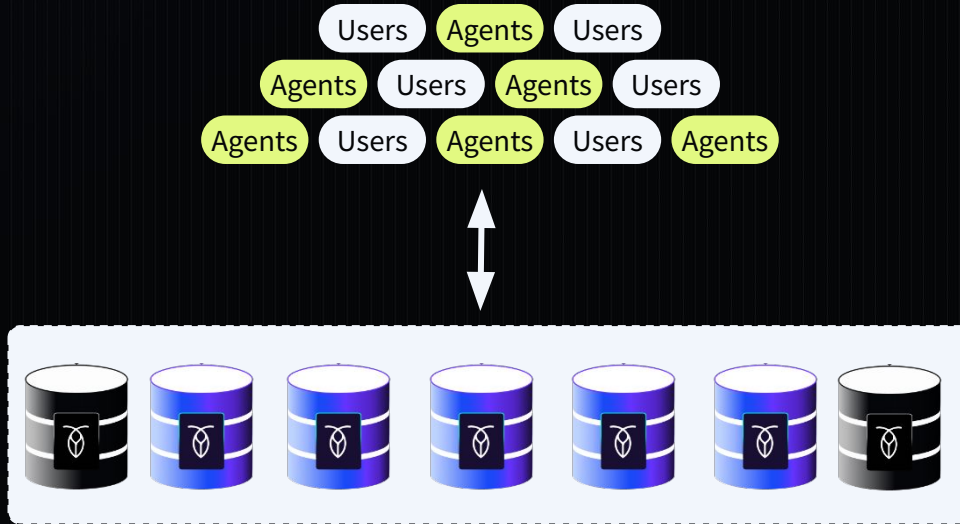
Build Once, Scale Users/Agents Forever



Build Once, Scale Users/Agents Forever



Build Once, Run forever



Business Continuity in the event of disasters

CockroachDB & DBOS are better together

DBOS makes your Workflows / AI agents reliable. CockroachDB makes the reliability indestructible.

Your workflows / agents never start over.

DBOS checkpoints every step of your agent's workflow. If anything crashes, it picks up exactly where it left off — no repeated work or lost progress.

Your data never goes down.

CockroachDB automatically survives server, data center, and entire region failures with zero data loss and zero manual intervention. Data is always there.

Your agents never act twice.

DBOS writes checkpoints in the same transaction as your business data. Operations happen exactly once — guaranteed by CockroachDB's ACID core.

Scale by adding machines.

Need more agents? Add CockroachDB nodes. No sharding, no re-architecture, and no migrations. Workflows scale right along with the underlying database.

One database for everything.

Transactional data, workflow state, and vector embeddings for semantic memory — all in CockroachDB. No separate state stores or sync pipelines.

Agents as first-class users.

Managed MCP Server gives agents a secure interface to data. DBOS plugs into OpenAI Agents SDK, Pydantic AI, LlamaIndex, Google ADK, and more.

Get started with CockroachDB & DBOS

Get started with CockroachDB Cloud
and get \$400 in free credits.



Visit cockroachlabs.cloud/signup

DBOS

Free open source durable workflow
execution library

<https://dbos.dev>

<https://github.com/dbos-inc>

Q&A