1. Core Problem Faced

We faced persistent 'not claimable for execution' errors.
Calls were being selected by scheduler or trigger but rejected during execution.
Root cause was not concurrency, but a broken and inconsistent state machine.

2. Root Cause Identified

Multiple code paths violated the same state transition rules.
Scheduler, manual trigger, and fakeDial were not aligned.
Some paths skipped CLAIMED state entirely.

3. Final Correct State Machine

CREATE → PENDING
SCHEDULER/TRIGGER → CLAIMED
EXECUTOR → IN_PROGRESS
PROVIDER EVENTS → COMPLETED | FAILED | PENDING (retry)

4. Why CLAIMED State Is Critical

CLAIMED establishes exclusive ownership.
Only one process can move a call from PENDING to CLAIMED.
Without CLAIMED, duplicate execution and race conditions occur.

5. Scheduler Fix

Scheduler must:
1. SELECT eligible PENDING calls
2. Attempt atomic UPDATE to CLAIMED
3. ONLY execute fakeDial if claim succeeds

Calling fakeDial without successful claim is a bug.

6. Trigger Fix

Manual trigger must behave exactly like scheduler.
It must:
- Update PENDING → CLAIMED
- Never jump directly to IN_PROGRESS
- Delegate execution to fakeDial

7. fakeDialer Fix

fakeDial must:
- Accept only CLAIMED calls
- Update CLAIMED → IN_PROGRESS
- Use result.rowCount for UPDATE validation
This guarantees executor correctness.

8. Database Truth Principle

Never trust JS if/else for concurrency decisions.
The database must decide state transitions using atomic UPDATEs.
If UPDATE affects 0 rows, the action must stop.

9. Retry Logic Learnings

Provider events decide the future, not the past.
no_answer + retries left → PENDING (reschedule)
no_answer + retries exhausted → FAILED
completed → COMPLETED

10. Inbound vs Outbound Separation

Inbound calls:
- Do not use scheduler
- Start directly in IN_PROGRESS
- End in COMPLETED

Outbound calls:
- Always go through PENDING → CLAIMED → IN_PROGRESS

11. Column Consistency Lesson

Schema and code must always match.
Mismatch between retries vs retry_count caused silent failures.
Schema evolution must update all code paths.

12. Route Design Lesson

Avoid overlapping route prefixes.
Clear separation like:
/webhooks/inbound
/provider/event
prevents debugging nightmares.

13. Dev Data Hygiene

Old corrupted states caused false errors.
In development, resetting state is normal and necessary.
Dirty data hides real bugs.

14. Universal Backend Lessons

- One owner per state transition
- One executor per job
- Atomic DB transitions
- State machines over ad-hoc flags

These lessons apply to job queues, payments, emails, and workflows.

15. Final Outcome

System is now:

- Concurrency-safe
- Deterministic
- Debuggable
- Production-grade

Most importantly:

The mental model is now correct.