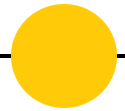
Online, real-time, multiplayer

PONG game

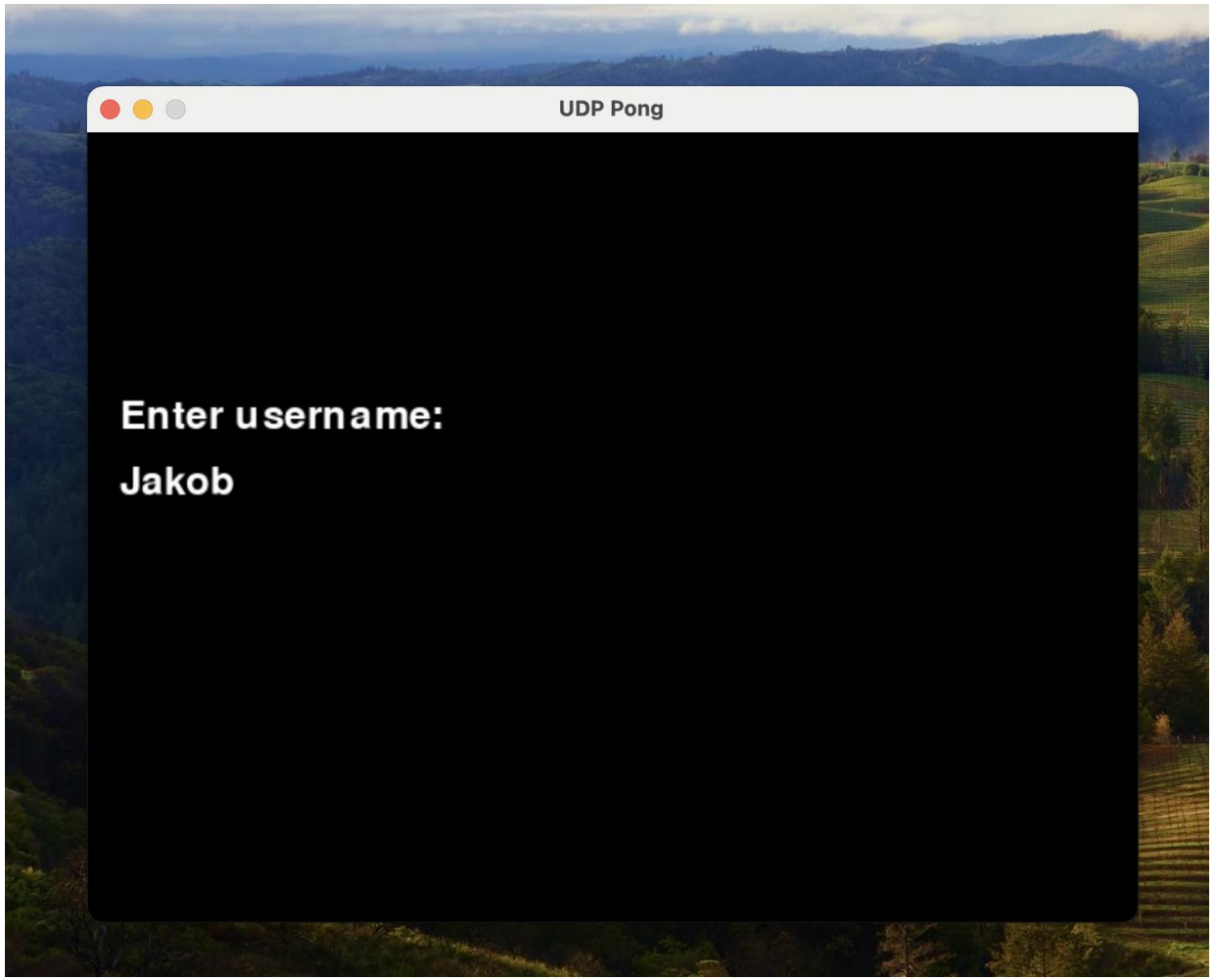


CS 2620 Final Project

Jakob de Raaij, Ulrik Unneberg

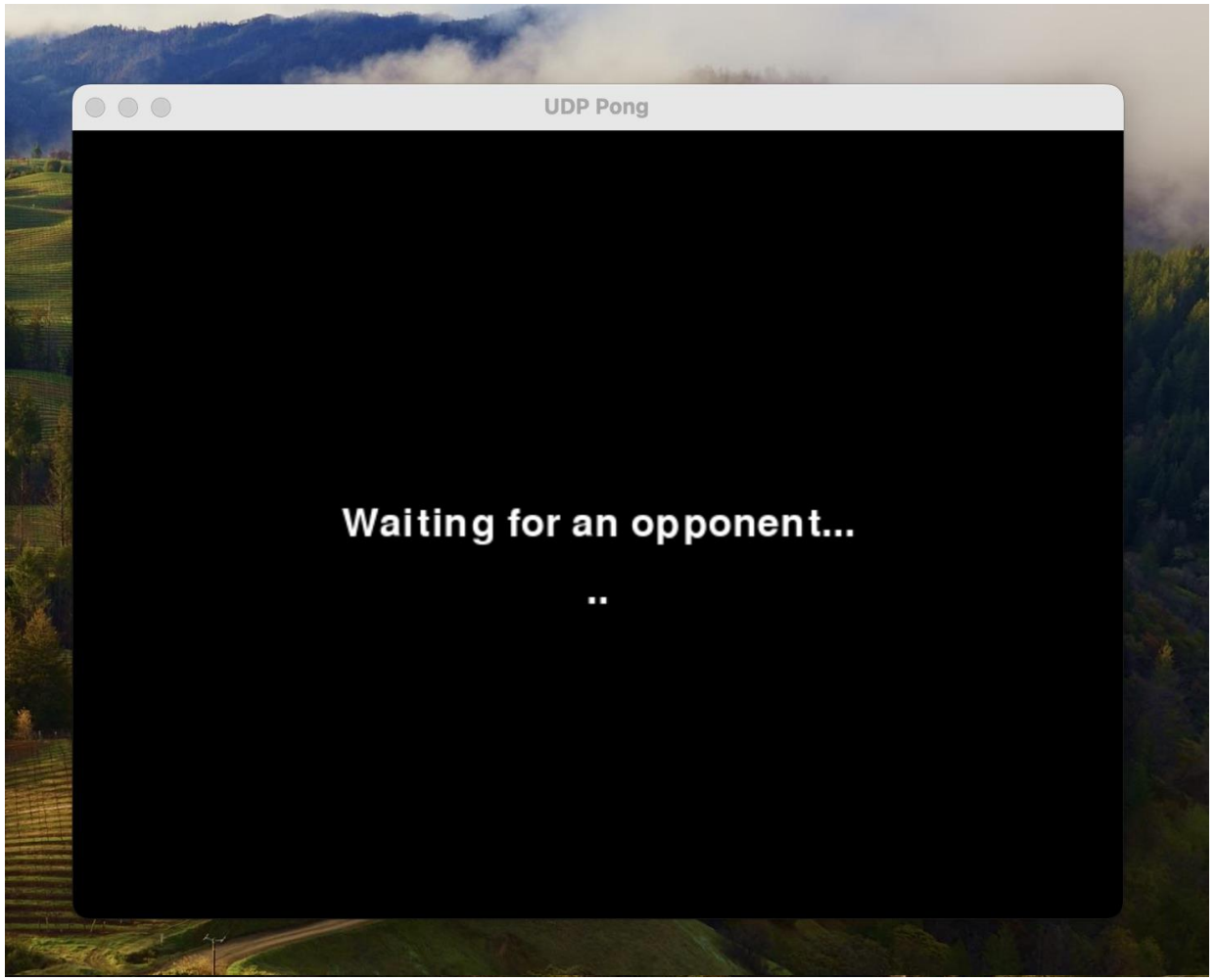
Application Demo

- Authentication



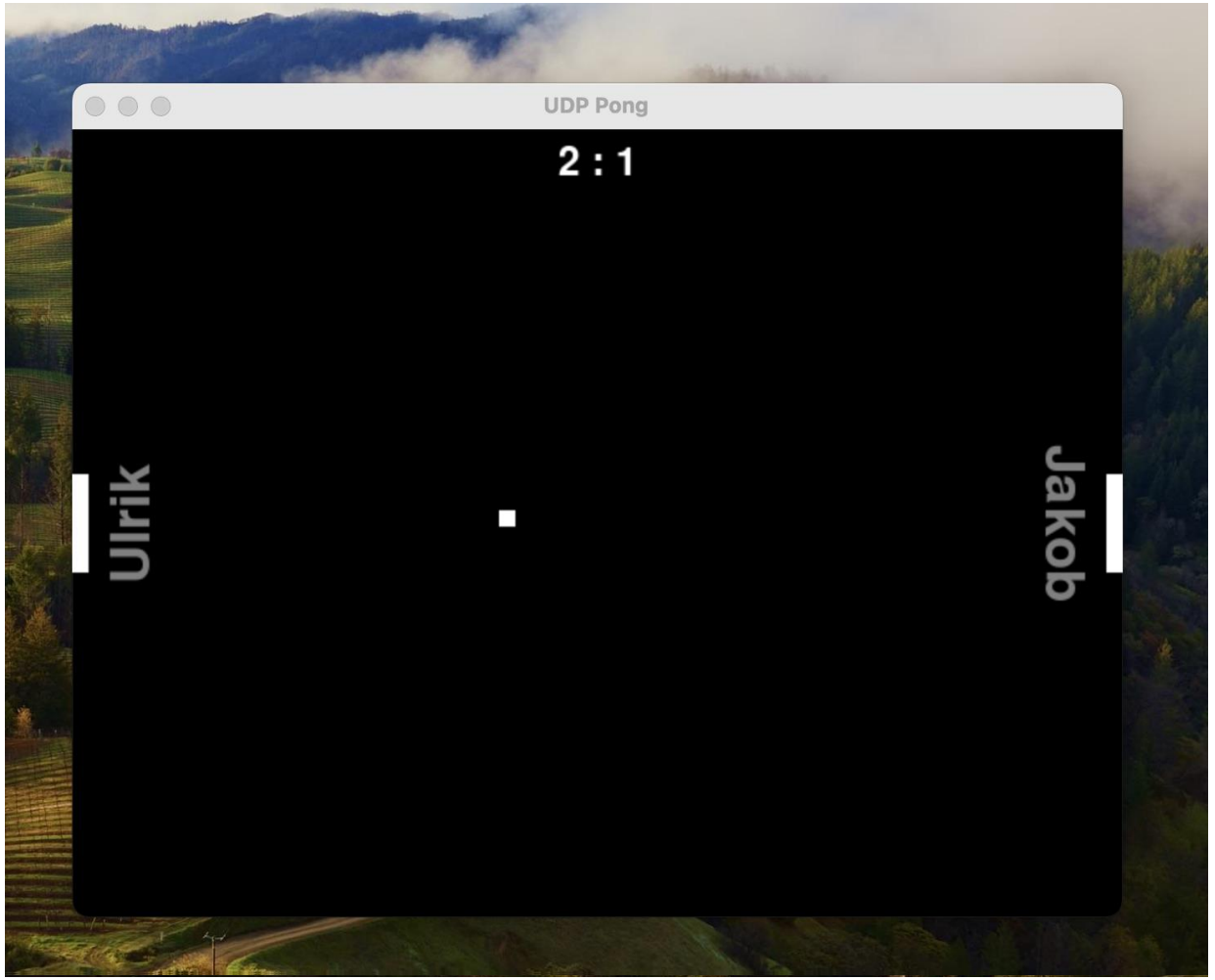
Application Demo

- Authentication
- Lobby system for matchmaking



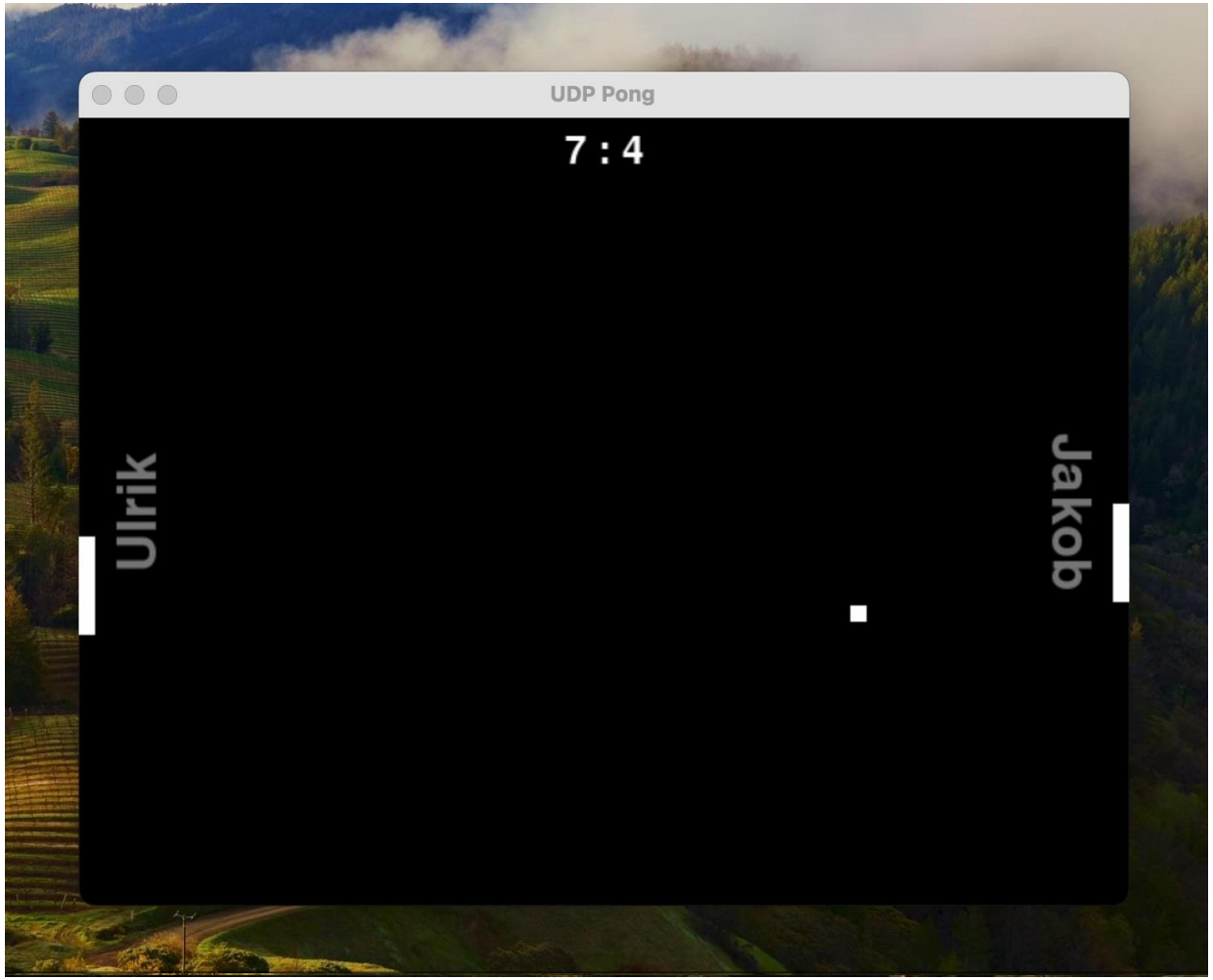
Application Demo

- Authentication
- Lobby system for matchmaking
- PONG game



Application Demo

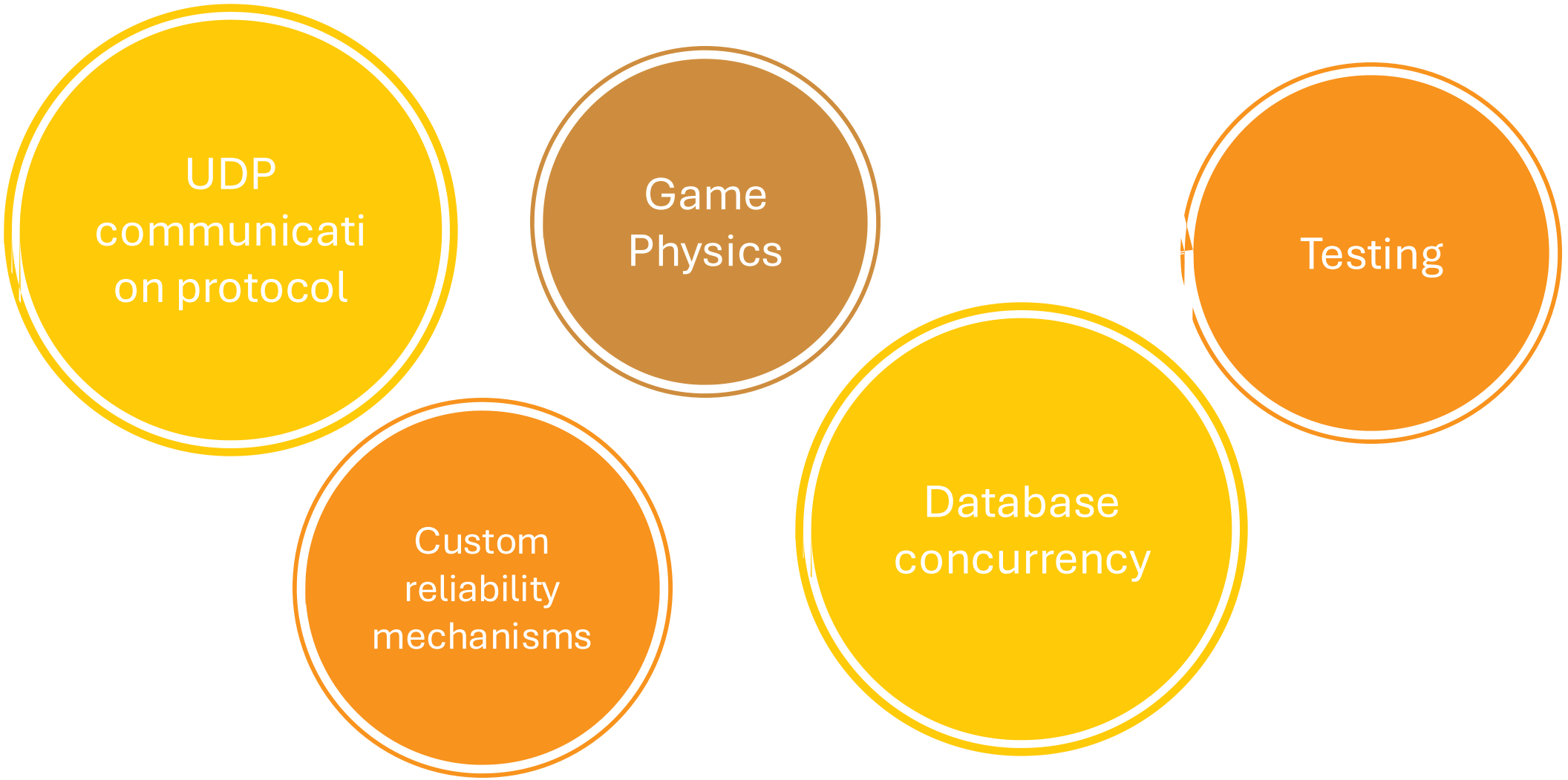
- Authentication
- Lobby system for matchmaking
- PONG game



Application Demo

- Authentication
- Lobby system for matchmaking
- PONG game
- Game statistics

Implementation Details



UDP Protocol

- Communication protocol for low-latency communication
- Compared to TCP:
  - NO connection is established
  - NO acknowledgment of received messages
  - NO ordering of messages
  - Very little error checking

→ Fast communication for game

→ doesn’t matter to much if single messages are lost

Reliability with UDP

Need to ensure reliability when needed

- **`Pulse`** to let server know client is alive
  - **`Pulse`** every 2 seconds, raise exception if none for 5 seconds
- **`Hello`/`Welcome`** protocol for initial handshake
  - Authenticated client sends **`Hello`** message, retries if no **`Welcome`** received
- Automatic reauthentication
  - In case of network issues, the client automatically reauthenticates

Client-side prediction

- For smooths game flow, client *predicts* game state
  - Updates with new package from server
- Server is authoritarian

Game Physics

- In general, stuck with ‘real’ physics

  Angle of incidence = angle of reflection

- Added randomness to increase fun (and protect against deterministic bots)
  - Starting angle is random (roughly $-30°$ to $30°$)
  - Slight change to reflection angle on each bounce (up to roughly $10°$)

Database Concurrency

- Using SQLite with Write-Ahead Logging
  - Clients can read from database while another client writes
  - Writes go to separate WAL-database
  - If WAL-db is locked, clients re-check periodically
- Periodically, commits get transferred to database
  - No concurrency issues since every user only accesses their ‘own’ data
- Problem: doesn’t scale across multiple machines, need different mechanism for server replication

Testing

- Unit and integration tests
  - Physics, bounces of ball
  - Scoring
  - Server communication
  - Lobby system
  - Unexpected behavior

- Database concurrency
  - 5 processes concurrently log wins and loses
  - WAL-logic achieves 70% reduced lock time


PONG