Load Testing Report — Belote

Overview

Under progressively higher concurrency, the system maintained correctness and graceful degradation. I identified the game-start hot spot at high load (p95 ~1.27s; start retries exhausted for some games)without cascading failures, and I defined concrete adaptations (queueing and horizontal scaling) to extend the scalability.

1. Objectives

- **Validate scalability and resilience** of the microservices architecture under rising concurrency.
- **Quantify latency and throughput** across low, medium, and high load.
- **Observe failure modes** (e.g., “lobby full”, game start contention) and confirm graceful degradation.
- **Assess readiness** for horizontal scaling and future adaptation.

2. Test Environment & Method

- **Tool:** k6 (local runs).
- **Scenarios:** constant VUs for short intervals; graceful stop enabled.
- **Workload per iteration (5 requests):**
    1. create lobby (expect **201**)
    2. join lobby (expect **200**)
    3. additional 3 requests including a **negative-path “lobby full”** assertion
- **Note on metrics:** k6 counts non-2xx/3xx as `http_req_failed`. The “lobby full” endpoint is expected to return a non-2xx; therefore ~**20% http_req_failed is expected** (1 of 5 requests per iteration).

3. Scenarios

- **Low:** 100 VUs for 10s
- **Medium:** 200 VUs for 30s
- **High:** 1000 VUs for 45s

4. Results Summary

Scenario	VUs	Duration	Iterations	HTTP Reqs	Avg Req Latency	p90	p95	http_req_failed	Notes
Low	100	10s	**4,248**	**21,240**	**27.29 ms**	35.25 ms	39.39 ms	**20.00%**	Expected (negative-path "lobby full")
Medium	200	30s	**9,404**	**47,020**	**107.95 ms**	173.24 ms	204.11 ms	**20.00%**	k6 high-cardinality warning triggered
High	1000	45s	**14,074**	**70,370**	**632.22 ms**	**1.04 s**	**1.27 s**	**20.00%**	High-cardinality warning; game-start contention observed

Throughput & Mix sanity check: `http_reqs / iterations = 5` for all runs → 5 requests per iteration as designed.
Check assertions: 100% passing for positive-path checks (`create 201`, `join 200`, and the explicit "lobby full" check).

5. Detailed Observations

5.1 Latency & Scalability

- **Low load:** Excellent latency (avg ~27 ms, p95 < 40 ms).
- **Medium load:** Latency increases proportionally (avg ~108 ms, p95 ~204 ms) but remains acceptable for interactive play.
- **High load:** Latency degrades (avg ~632 ms, **p95 ~1.27 s**). Still serviceable, but this is your **current scalability boundary** before user experience might suffer.

LO3 link — Scalability: The system scales predictably until high-load, where response times cross ~1s at the 95th percentile. This demonstrates horizontal scalability potential and identifies a clear performance envelope for capacity planning.

5.2 Resilience & Failure Modes

- **Expected "failures":** `http_req_failed = 20%` corresponds to one intentional negative-path request per iteration ("lobby full"), confirming correct boundary behavior under contention.
- **Game start contention (High load):** "Many games were not started" even after 5 retries. **Critically:** *no cascading failures or service crashes observed*; the system degraded gracefully.

LO3 link — Resilience & Adaptation: The architecture handled hot paths and rejection paths without instability, evidencing **graceful degradation** and **fault isolation** between services.

5.3 Metrics Cardinality

- k6 warnings at Medium/High: **100k–200k unique time series**. This is due to **high-cardinality tags** (likely unique IDs in URLs/tags).

- **Impact:** Increased memory usage and noisier dashboards.
- **Action:** Group URLs, avoid embedding unique IDs in tags, use k6 `thresholds` and `groups` with low-cardinality labels.

LO3 link — Operability (quality attribute): Observability practices are in place; improvements to metric hygiene will make scaling the monitoring stack easier.

6. Potential Improvements

1. Increase **replicas** for the lobby/game services under load (horizontal scaling)
2. Scale **game start** logic via **work queues** (NATS JetStream).
3. Track **domain metrics**: lobby creation rate, join success, game-start success/fail, queue depth, retry counts, lock wait times (currently only being logged, not followed or visualized).