Các Giai Đoạn Load Test (betting)

Giai đoạn 1: Ramp-up (Tăng dần từ 500k TPS lên 1M TPS) (5’)

1. **Mục đích**
    - Để hệ thống làm nóng (warm-up) và tìm ngưỡng ổn định ban đầu mà không gây “sốc” ngay lập tức.
2. **Kịch bản**
    - Bắt đầu gửi **500.000 giao dịch/giây (TPS)**.
    - Mỗi **1 phút**, tăng thêm **100.000 TPS**.
    - Tiếp tục tăng dần cho đến khi **chạm mốc 1.000.000 TPS**.
3. **Theo dõi**
    - **CPU, RAM**, và **network I/O** ở từng mốc TPS (500k, 600k, 700k...).
    - **Latency** trung bình, **error rate**.
    - Nếu phát sinh lỗi vượt ngưỡng (error rate quá cao), cần dừng hoặc chậm lại.

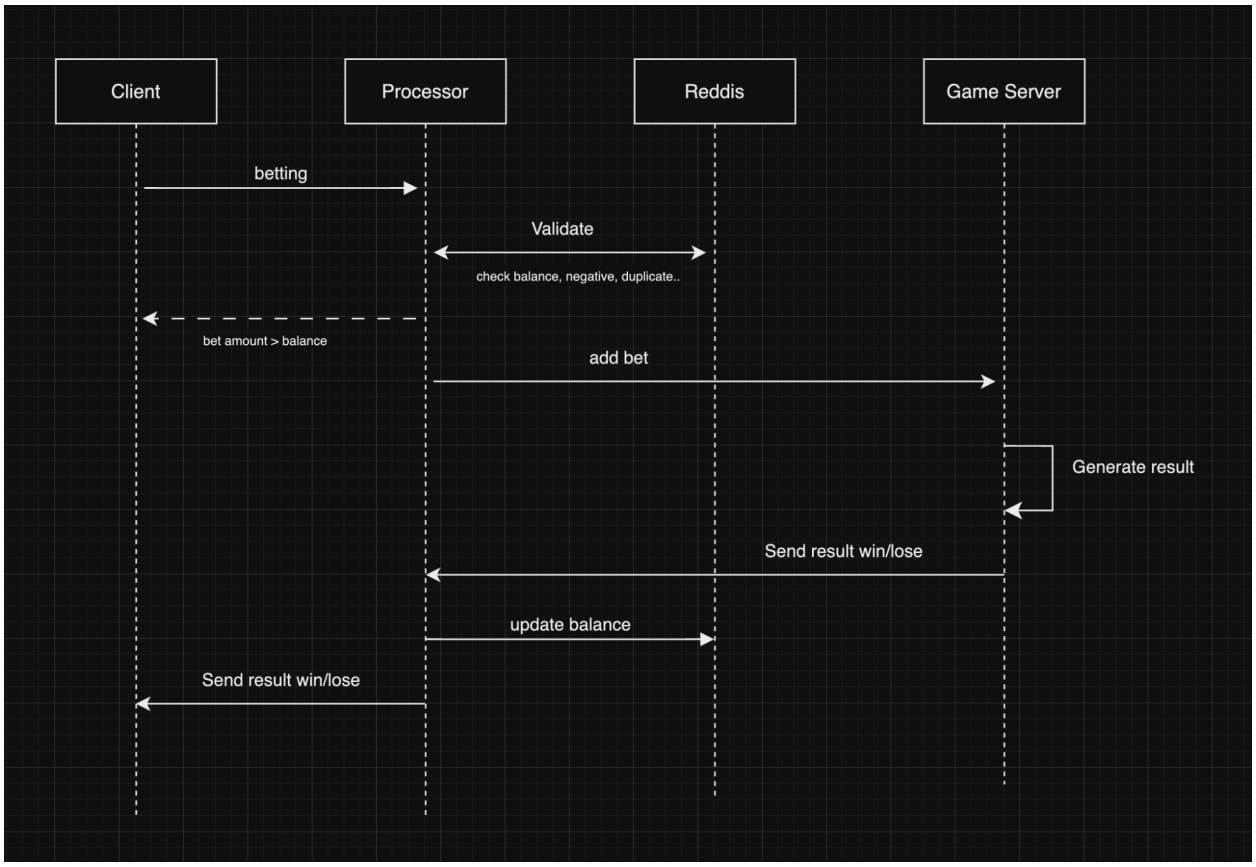
Giai đoạn 2: Steady State (Duy trì ở 1M TPS) (5’)

1. **Mục đích**
    - Kiểm tra khả năng chịu tải **liên tục** ở mức **1M TPS**.
    - Đánh giá độ **ổn định** của hệ thống trong khoảng thời gian nhất định 5 phút.
2. **Kịch bản**
    - Sau khi đạt 1M TPS, **giữ tải** ở mức này trong **5 phút** .
    - Ghi nhận chỉ số hiệu năng(latency, error rate, resource usage).
3. **Theo dõi**
    - **TPS trung bình** và **dao động** (nếu có).
    - **Resource usage** (CPU có gần 100% không, RAM còn dư bao nhiêu?).
    - **Stability**: TPS có bị sụt giảm bất thường hoặc tăng đột biến lỗi?

Giai đoạn 3: (Tuỳ chọn) Stress Test Vượt Ngưỡng (5’)

1. **Mục đích**
    - Tìm “điểm phá vỡ” (breaking point) của hệ thống.
    - Đánh giá phản ứng khi vượt mốc 1M TPS.
2. **Kịch bản**
    - Tiếp tục tăng TPS lên trên 1M (mỗi phút tăng 50k-100k TPS) cho đến khi hệ thống quá tải.
    - Ghi nhận **thời điểm** và **dấu hiệu** quá tải (error rate tăng mạnh, latency vượt ngưỡng, CPU 100%...).
3. **Theo dõi**
    - **Cách hệ thống degrade**: dừng xử lý, trả lỗi , hay phản hồi chậm dần?

- **Khả năng phục hồi** (nếu giảm tải lại xuống 1M hoặc 800k TPS, liệu hệ thống có về trạng thái ổn định?).



Node 1: for load test
Node 2: for services processing

Balance Consistency: Validate that the **bet amount does not exceed the balance**.

Required fields: `user_id`, `transaction_id`, `amount`, `action` [bet, win, lose].

Create 100k accounts

Deposit money to above accounts

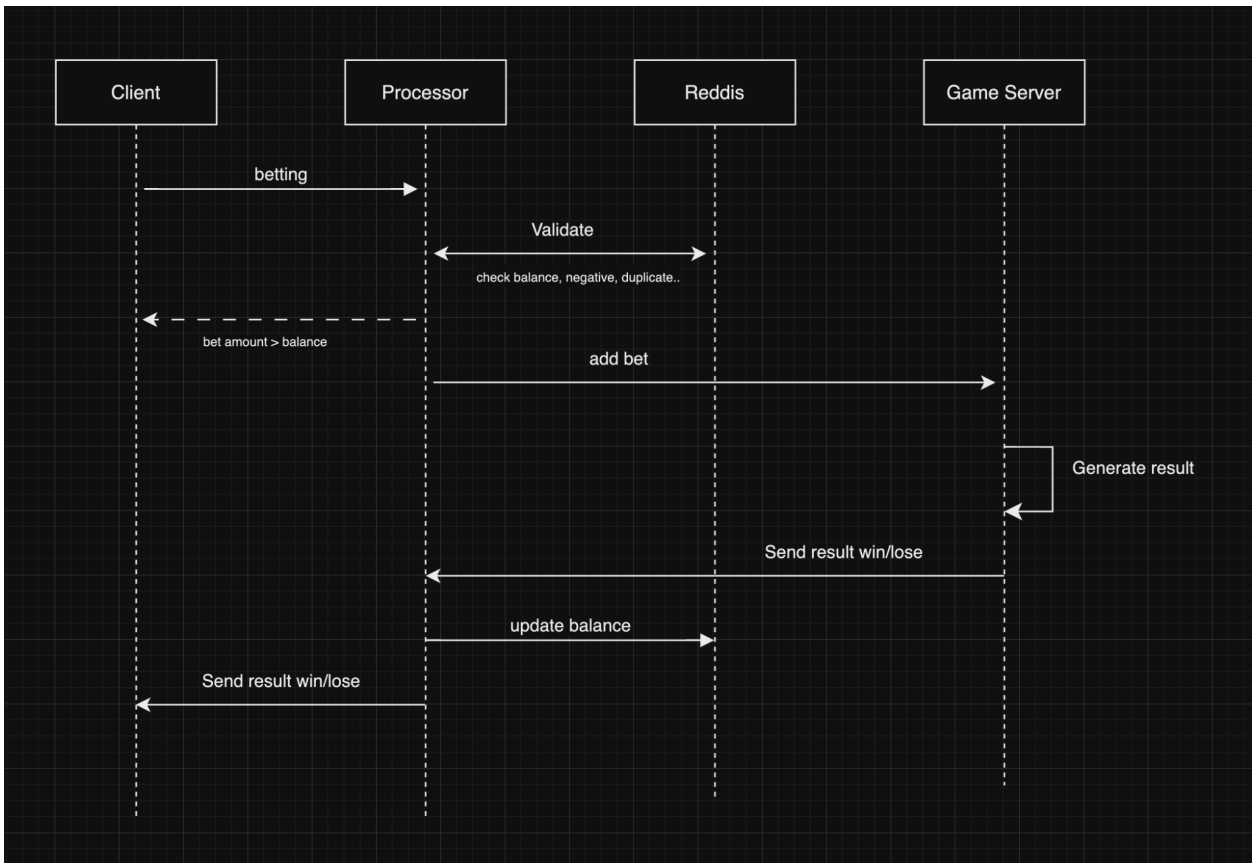
** Betting with random amount (1,5,10) within above accounts

Check balance update after betting -> balance consistence

Đánh Giá Cuối Cùng (5’)

1. **Hiệu năng**
    - **TPS** (trung bình, đỉnh). Có đạt được mục tiêu (700k, 1M, ...) không?
    - **Latency**: trung bình, P95, P99.
    - **Error rate**: Đã ở mức chấp nhận được (< 1%)?
2. **Tính nhất quán (Balance Consistency)**
    - Số tài khoản kiểm tra vs. số tài khoản lệch.
    - Sai lệch **balance** cao hay thấp?
    - Giao dịch bị mất, bị trùng, hay lỗi logic nào?
3. **Tài nguyên**
    - CPU, RAM, DB connections, network I/O.
    - Phát hiện bottlenecks (nếu có).
4. **Tính chịu lỗi** (nếu giả lập):
    - Hệ thống có phục hồi nhanh khi node xử lý bị tạm ngắt không?
    - Bao nhiêu giao dịch bị mất?

ENGLISH VERSION



Node 1: for load test
Node 2: for services processing (processor, redis, game server)

Balance Consistency: Validate that the **bet amount does not exceed the balance**.

Required fields: `user_id`, `transaction_id`, `amount`, `action` [bet, win, lose].

> Create 100k accounts (script to create account, script to check the account has been created)
>
> Deposit money to above accounts (script to deposit , script to check the result of deposit for 100k accounts) 100$ each account.
>
> Betting with amount (1$) within above accounts (each account have at least 5 transactions, have a script to check the transaction history of accounts)
>
> Check balance update after betting -> balance consistence => have a query to check the betting history of accounts

Load test in betting

Phase 1: Ramp-up (Increasing from 500k TPS to 1M TPS) (5')

Objective

- Warm up the system and identify the initial stable threshold without causing an immediate "shock."

Scenario

1. Start by sending **500,000 Transactions Per Second (TPS)**.
2. Increase by **100,000 TPS every minute**.
3. Gradually ramp up until reaching **1,000,000 TPS**.

Monitoring

- **System resources**: CPU, RAM, and network I/O at each TPS milestone (500k, 600k, 700k...).
- **Performance metrics**: Average latency, error rate.
- If error rates exceed acceptable thresholds (5%), slow down or stop the process.

Phase 2: Steady State (Maintaining 1M TPS) (5’)

Objective

- Test the system's capability to sustain continuous load at **1M TPS**.
- Evaluate stability over a fixed period of **5 minutes**.

Scenario

1. After reaching **1M TPS**, maintain this load for **5 minutes**.
2. Record performance metrics (latency, error rate, resource usage).

Monitoring

- **TPS consistency**: Average TPS and any fluctuations.
- **Resource usage**: Is CPU close to 100%? How much RAM is available?
- **Stability**: Are there abnormal TPS drops or error spikes?

Phase 3 (Optional): Stress Test Beyond Threshold (5’)

Objective

- Identify the system's **breaking point**.
- Assess the system’s behavior when exceeding **1M TPS**.

Scenario

1. Gradually increase TPS beyond **1M TPS** (adding 100k-300k TPS per minute) until the system becomes overloaded.
2. Record the overload point and related indicators (high error rates, excessive latency, 100% CPU usage).

Monitoring

- **System degradation**: Does the system stop processing, return errors, or slow down gradually?
- **Recovery capability**: If the load is reduced to **1M TPS** or **800k TPS**, does the system return to a stable state?

Final Assessment

Performance (5')

- **TPS**: Average and peak TPS. Did the system meet the targets (700k, 1M, etc.)?
- **Latency**: Average, **P95**, **P99**.
- **Error rate**: Was it within acceptable limits (< 1%)?

Consistency (Balance Consistency)

- Number of accounts checked vs. number of discrepancies.
- Severity of balance discrepancies.
- Lost, duplicate, or logical transaction errors.

Resources

- CPU, RAM, DB connections, network I/O.
- Detection of bottlenecks (if any).

Fault Tolerance (Simulated Failures)

- Does the system recover quickly when a processing node is temporarily disconnected?
- How many transactions were lost?