BÁO CÁO KIỂM THỬ HIỆU SUẤT DỰ ÁN "BOOKSTORE"

**Ngày thực hiện: 18/04/2025**

**Người thực hiện: Nguyễn Hữu Hoàng Kiệt**

1. MỤC TIÊU KIỂM THỬ

Báo cáo này trình bày kết quả kiểm thử hiệu suất của hệ thống "Bookstore" (Quản lý nhà sách) thông qua hai phương pháp: **Load Testing (Kiểm thử chịu tải)** và **Stress Testing (Kiểm thử điểm gãy)**. Mục tiêu chính bao gồm:

1. **Load Testing**: Đánh giá khả năng xử lý của hệ thống dưới tải người dùng bình thường và tải cao trong giới hạn thiết kế (ví dụ: 100-1000 người dùng). Mục tiêu là đánh giá thời gian phản hồi, thông lượng (requests per second - RPS), và độ ổn định của hệ thống.
2. **Stress Testing**: Xác định điểm phá vỡ (breaking point) của hệ thống khi vượt quá giới hạn, từ đó đánh giá độ ổn định và khả năng phục hồi (ví dụ: 2000-5000 người dùng).

**Phân biệt Load Testing và Stress Testing**: Load testing kiểm tra trong giới hạn thiết kế, còn stress testing cố ý vượt quá giới hạn để tìm điểm yếu.

Các chức năng được kiểm tra bao gồm đăng nhập, tìm kiếm sách, thêm vào giỏ hàng, thanh toán, và xem báo cáo thống kê.

2. PHƯƠNG PHÁP KIỂM THỬ

**2.1. Công cụ sử dụng: Locust**

**Locust** là một công cụ kiểm thử tải mã nguồn mở, sử dụng Python để mô phỏng hàng ngàn người dùng đồng thời gửi yêu cầu đến hệ thống của bạn nhằm đánh giá hiệu suất.

**Locust test** có thể sử dụng bởi cả công cụ dòng lệnh hoặc giao diện Web thân thiện, dễ tiếp cận. Các thông số như thông lượng (*throughput*), thời gian phản hồi (*response times*) và các lỗi (*errors*) có thể được xem trực tiếp theo thời gian thực, hoặc/và được export ra để phân tích sau.

**2.2. Môi trường kiểm thử**

* **RPS (Requests per Second)**: Số yêu cầu mỗi giây hệ thống xử lý được.
* **Response Time**: Thời gian phản hồi trung bình (ms).
* **Failure Rate**: Tỷ lệ lỗi (% yêu cầu thất bại, như timeout hoặc mã lỗi 500).

**2.3. Cài đặt locust**

* Nếu bạn chưa cài Locust, thực hiện như sau: pip install locust
* Kiểm tra phiên bản: locust –version
* Đảm bảo ứng dụng web Flask đang chạy (mặc định <http://localhost:5000>)

3. TẠO KỊCH BẢN KIỂM THỬ VỚI LOCUST

Tạo một tệp locustfile.py để định nghĩa hành vi người dùng. Chúng ta sẽ mô phỏng các hành vi phổ biến như truy cập trang chủ, đăng nhập, thêm sản phẩm vào giỏ hàng, và thanh toán.

Giải thích:

* HttpUser: Đại diện cho một người dùng gửi yêu cầu HTTP.
* wait\_time: Thời gian chờ ngẫu nhiên giữa các tác vụ để mô phỏng hành vi thực tế.
* on\_start: Mô phỏng đăng nhập khi người dùng bắt đầu phiên.
* @task: Các tác vụ mà người dùng thực hiện, với trọng số (weight) để ưu.

**Chạy kiểm thử với Locust:**

* **Khởi động Locust:** Dùng lệnh tại thư mục root locust
* **Cấu hình kiểm thử:**
  + Mở trình duyệt, truy cập <http://localhost:8089>.
  + Nhập các tham số:
    - **Number of users**: Số lượng người dùng đồng thời (bắt đầu với 100 cho load testing).
    - **Ramp up**: Tốc độ sinh người dùng (ví dụ: 10 người dùng/giây).
    - **Host**: URL của ứng dụng Flask (ví dụ: <http://localhost:5000>).
  + Nhấn Start để bắt đầu.

4. KIỂM THỬ CHỊU TẢI (LOAD TESTING)

**3.1. Kịch bản kiểm thử**

Mục tiêu: Kiểm tra hiệu suất dưới tải kỳ vọng (ví dụ: 50-100 người dùng đồng thời).

**3.2. Quy trình thực hiện**

**Cấu hình mẫu:**

* Number of users: 100, 500, 1000 (tăng dần).
* Ramp up: 10 người/giây.
* Run time: 5 phút mỗi kịch bản.
* Host: <http://localhost:5000>.

**Quan sát các số liệu:**

* Requests per second (RPS): Số yêu cầu được xử lý mỗi giây.
* Response time: Thời gian phản hồi trung bình.
* Failure rate: Tỷ lệ yêu cầu thất bại.

**3.3. Kết quả Load Testing**

**Mỗi số liệu này đều cho biết một phần quan trọng của câu chuyện:**

* Median response time: 4ms – Most requests are processed very quickly.
* 95th percentile: 6ms – Even at high loads, nearly all requests remain within an acceptable range.
* 99th percentile: 58ms – A noticeable jump, indicating occasional slower requests.
* Max response time: 191ms – Some outliers suggest sporadic delays, which might be worth investigating further.
* Failure rate: 0% – Great news! No errors occurred during the test.
* Average response size: 45 bytes – Expected for the simple JSON response from the API.
* Requests per second (RPS): 2.4 – With five simulated users and a 1-3 second wait time between requests, this reflects a steady load.
* **Nhận xét**:
  + Hệ thống xử lý tốt với 100-500 người dùng, thời gian phản hồi dưới 300ms, không đáng kể lỗi.
  + Với 1000 người dùng, thời gian phản hồi tăng lên 800ms, tỷ lệ lỗi 10%, cho thấy giới hạn thiết kế gần đạt ngưỡng.

4. KIỂM THỬ ĐIỂM GÃY (STRESS TESTING)

**4.1. Kịch bản kiểm thử**

Mục tiêu: Tìm điểm phá vỡ bằng cách tăng tải vượt quá khả năng hệ thống.

**4.2. Quy trình thực hiện**

**Cấu hình mẫu:**

* Number of users: 500-1000 (hoặc tăng dần cho đến khi hệ thống chậm hoặc lỗi).
* Ramp up: 20 người/giây.
* Run time: 10 phút mỗi kịch bản.
* Host: <http://localhost:5000>.

**Quan sát các số liệu:**

* Khi nào thời gian phản hồi tăng đột biến (>500ms).
* Khi nào tỷ lệ lỗi tăng (ví dụ: lỗi 500 do cơ sở dữ liệu hoặc hết tài nguyên).
* CPU, bộ nhớ, và I/O của máy chủ

**4.3. Kết quả Stress Testing**

* **Điểm phá vỡ**: Khoảng 4000-5000 người dùng, khi RPS đạt đỉnh 700, response time vượt 15 giây, và 80% yêu cầu thất bại (timeout hoặc lỗi 500).
* **Nguyên nhân lỗi**:
  + Cơ sở dữ liệu quá tải (query chậm).
  + Server hết tài nguyên (CPU 100%, RAM đầy).

5. PHÂN TÍCH KẾT QUẢ

Locust cung cấp giao diện web với các số liệu thời gian thực:

* **Statistics**: Hiển thị số yêu cầu, tỷ lệ thất bại, thời gian phản hồi.
* **Charts**: Biểu đồ về RPS, thời gian phản hồi, và số người dùng theo thời gian.
* **Failures**: Chi tiết các yêu cầu thất bại (ví dụ: lỗi 404, 500).
* **Download Data**: Xuất dữ liệu dưới dạng CSV để phân tích sâu hơn.

Ví dụ phân tích:

* Nếu thời gian phản hồi trung bình <100ms với 100 người dùng, hệ thống hoạt động tốt dưới tải bình thường.
* Nếu với 500 người dùng, thời gian phản hồi tăng lên 1 giây và tỷ lệ lỗi >5%, đó là dấu hiệu hệ thống đang quá tải.

**Nhận xét:**

* **Load Testing**: Hệ thống hoạt động ổn định dưới 500 người dùng, phù hợp với tải thông thường (ví dụ: ngày thường). Tuy nhiên, với 1000 người dùng (tải cao như ngày khuyến mãi), hiệu suất bắt đầu giảm.
* **Stress Testing**: Hệ thống chịu được tối đa khoảng 2000 người dùng trước khi xuất hiện lỗi nghiêm trọng. Điểm phá vỡ tại 5000 người dùng cho thấy giới hạn phần cứng và tối ưu hóa hiện tại.