TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CÔNG CỤ KIỂM THỬ LOCUST

MÔN HỌC: KIỂM THỬ VÀ ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

Giảng viên: ThS. Nguyễn Thu Trang

Sinh viên 1: Nguyễn Xuân Thịnh – 23020709

Sinh viên 2: Hoàng Duy Thịnh – 23020708

Sinh viên 3: Lê Duy Khánh Toàn – 23020702

Sinh viên 4: Đặng Đình Khang – 23020709

HÀ NỘI - 07/2025

Mục lục

1. Tổng quan

Trong quá trình phát triển phần mềm, kiểm thử phần mềm là một quá trình quan trọng trong việc phát triển phần mềm, nhằm đảm bảo rằng phần mềm hoạt động đúng theo yêu cầu và không có lỗi. Quá trình này bao gồm việc thực hiện các hoạt động kiểm tra để phát hiện và sửa chữa các lỗi trong phần mềm trước khi sản phẩm được phát hành đến người dùng cuối.

Công cụ kiểm thử phần mềm chính là trợ thủ đắc lực cho các nhà phát triển trong việc tự động hóa, tối ưu hóa quá trình kiểm tra, tiết kiệm thời gian, chi phí và nâng cao hiệu quả. Công cụ kiểm thử giảm thiểu khả năng xảy ra lỗi do con người, đảm bảo kết quả kiểm thử nhất quán và chính xác hơn; có thể chi phí đầu tư ban đầu cho công cụ kiểm thử cao, nhưng về lâu dài, nó giúp giảm chi phí phát sinh do lỗi phần mềm và các sự cố liên quan.

Trong bài báo cáo này, nhóm em tập trung vào công cụ kiểm thử Locust-công cụ kiểm thử hiệu năng được viết bằng Python.

1.1 Định nghĩa kiểm thử

Kiểm thử phần mềm là quá trình thực hiện các hành động có kế hoạch và có hệ thống để kiểm tra một phần mềm, nhằm xác định chất lượng của phần mềm đó. Mục tiêu của kiểm thử phần mềm là phát hiện lỗi, đảm bảo chất lượng và xác nhận rằng phần mềm đáp ứng các yêu cầu đã đặt ra.

1.2 Vai trò của kiểm thử

Kiểm thử phần mềm đóng vai trò quan trọng trong việc đánh giá chất lượng và là hoạt động chủ chốt trong việc đảm bảo chất lượng cao của sản phẩm phần mềm trong quá trình phát triển. Thông qua chu trình "kiểm thử -tìm lỗi - sửa lỗi" ta hy vọng chất lượng của sản phẩm phần mềm sẽ được cải tiến. Mặt khác, thông qua việc tiến hành kiểm thử mức hệ thống trước khi cho lưu hành sản phẩm, ta biết được sản phẩm của ta tốt ở mức nào. Vì thế, nhiều tác giả đã mô tả việc kiểm thử phần mềm là một quy trình kiểm chứng để đánh giá và tăng cường chất lượng

của sản phẩm phần mềm. [1] (N. H. Pham, A. H. Truong, and V. H. Dang, in GIÁO TRÌNH KIỂM THỬ PHẦN MỀM, 2014.)

1.3 Một số thuật ngữ liên quan

2. Công cụ kiểm thử Locust

2.1 Tổng quan

Locust là một công cụ kiểm thử tải mã nguồn mở được viết bằng Python, dùng để đánh giá **hiệu năng và khả năng chịu tải của hệ thống**. Theo *tài liệu chính thức của Locust*, công cụ này cho phép người dùng mô phỏng hàng nghìn hoặc hàng triệu người dùng ảo truy cập vào website hoặc API để kiểm tra phản ứng của hệ thống dưới các mức tải khác nhau.

Điểm đặc biệt của Locust là cho phép **viết kịch bản kiểm thử (test scenario)** bằng Python, nhờ đó người kiểm thử có thể mô phỏng hành vi thực tế của người dùng, chẳng hạn như: truy cập trang chủ, đăng nhập, tìm kiếm sản phẩm, thêm vào giỏ hàng hoặc gửi yêu cầu API.

Khác với nhiều công cụ truyền thống như JMeter hoặc LoadRunner, Locust **nhẹ, linh hoạt và dễ mở rộng**, đồng thời có **giao diện web trực quan** để theo dõi kết quả kiểm thử theo thời gian thực. Ngoài ra, nó hỗ trợ **chế độ phân tán (distributed mode)**, cho phép chạy nhiều worker trên nhiều máy để kiểm thử quy mô lớn.

2.2 Các tính năng

Theo tài liệu chính thức và cộng đồng người dùng Locust, công cụ này sở hữu các nhóm tính năng nổi bật sau:

2.2.1 Mô phỏng người dùng thực tế

- Các hành vi của người dùng được định nghĩa thông qua các class kế thừa từ HttpUser hoặc User.
- Cho phép sử dụng hàm @task để xác định hành vi cụ thể và trọng số cho từng tác vụ.
- Mỗi người dùng ảo thực thi các hành vi này theo chu kỳ, mô phỏng hoạt động thực tế của người thật.

2.2.2 Kiểm thử tải và hiệu năng

- Đo **thời gian phản hồi (response time)**, **số lượng request/giây**, và **tỷ lệ lỗi**.
- Giúp phát hiện điểm nghẽn (bottleneck) của hệ thống khi số lượng người dùng tăng dần.
- Hỗ trợ giới hạn tốc độ tải hoặc đặt số lượng người dùng tăng theo thời gian (spawn_rate).

2.2.3 Giao diện web trực quan

- Giao diện tại địa chỉ http://localhost:8089 giúp điều chỉnh số người dùng, tốc độ sinh người dùng, và quan sát kết quả kiểm thử theo thời gian thực.
- Hiển thị thống kê như trung bình thời gian phản hồi, phần trăm lỗi, độ lệch chuẩn, và phân phối phần trăm phản hồi (percentiles).

2.2.4 Chạy phân tán

- Cho phép chạy theo mô hình **Master-Worker**, trong đó Master điều phối và tổng hợp kết quả từ nhiều Worker.
- Hữu ích khi cần kiểm thử hệ thống quy mô lớn với hàng trăm nghìn người dùng ảo.

2.2.5 Tích hợp và mở rộng

- Dễ dàng tích hợp vào pipeline CI/CD (Jenkins, GitHub Actions, GitLab CI).
- Có thể ghi log, xuất kết quả sang CSV hoặc gửi đến hệ thống giám sát như Grafana, Prometheus, InfluxDB.
- Hỗ trợ kiểm thử không chỉ HTTP mà còn WebSocket, MQTT, gRPC, và các giao thức tuỳ chỉnh thông qua lớp User.

2.3 Mục tiêu

Các mục tiêu chính của việc sử dụng Locust trong kiểm thử phần mềm bao gồm:

- **Đánh giá hiệu năng hệ thống:** Xác định khả năng chịu tải của hệ thống, tốc độ phản hồi và độ ổn định khi có nhiều người dùng đồng thời.
- **Phát hiện điểm nghẽn (bottlenecks):** Xác định thành phần gây ra độ trễ, như cơ sở dữ liệu, API chậm, hoặc quá tải CPU.
- **Tối ưu hóa tài nguyên:** Dựa trên kết quả kiểm thử, nhóm phát triển có thể tối ưu kiến trúc, cân bằng tải, hoặc nâng cấp phần cứng.
- **Đảm bảo độ tin cậy:** Kiểm thử khả năng phục hồi (resilience) và hành vi hệ thống trong các tình huống cực đoan.
- **Tự động hóa kiểm thử:** Tích hợp Locust vào quy trình CI/CD giúp đảm bảo hệ thống luôn duy trì hiệu năng ổn định qua các bản phát hành.

3. Đánh giá công cụ

4. So sánh công cụ Locust

5. Thực nghiệm

6. Tổng kết