1 Giới thiệu chung về Locust

Nền tảng kiến trúc của Locust dựa trên cơ chế xử lý đồng thời điều khiển bằng sự kiện (event-driven), sử dụng các coroutine cực nhẹ gọi là greenlet. Điều này cho phép một máy duy nhất có thể mô phỏng hàng ngàn người dùng đồng thời mà không tiêu tốn nhiều tài nguyên, một ưu điểm lớn so với các công cụ dựa trên luồng truyền thống. Để đánh giá đúng vị thế của Locust, nhóm em sẽ so sánh trực tiếp nó với ba công cụ phổ biến khác trong ngành: Apache JMeter, K6, và Gatling.

2 So sánh chi tiết

2.1 Locust vs. Apache JMeter

Apache JMeter là một trong những công cụ kiểm thử hiệu năng lâu đời và phổ biến nhất, nổi bật với giao diện đồ họa (GUI) trực quan cho phép người dùng xây dựng kịch bản mà không cần viết nhiều mã. Cuộc đối đầu giữa Locust và JMeter là sự so sánh giữa một bên là "tests-as-code" linh hoạt cho nhà phát triển và một bên là "GUI-driven" dễ tiếp cận cho người không chuyên về lập trình.

extbfBảng 4.1: So sánh Locust và Apache JMeter

>p3.2cm	>p5.7cm	>p5.7cm
extbfĐặc điểm	**Locust**	**Apache JMeter**
Ngôn ngữ chính	Python	Java (với Groovy/BeanShell)
Mô hình cốt lõi	Tests-as-Code	Kế hoạch kiểm thử qua GUI
Mô hình đồng thời	Dựa trên sự kiện (Greenlets)	Mỗi luồng một người dùng
Giao diện chính	Web UI & Dòng lệnh (CLI)	Giao diện đồ họa (GUI) & CLI
Báo cáo	Web UI thời gian thực	Báo cáo HTML chi tiết sau kiểm thử
Điểm mạnh chính	Linh hoạt, hiệu quả tài nguyên, dễ tích hợp CI/CD	Hỗ trợ nhiều giao thức, cộng đồng lớn, dễ bắt đầu không cần code

Nhận xét:
Sự khác biệt lớn nhất nằm ở kiến trúc. Mô hình "mỗi luồng một người dùng" của JMeter tiêu tốn nhiều tài nguyên hơn đáng kể so với mô hình dựa trên sự kiện của Locust. Do đó, Locust có khả năng mô phỏng số lượng người dùng đồng thời lớn hơn nhiều trên cùng một phần cứng. JMeter phù hợp cho các đội ngũ QA truyền thống hoặc khi cần kiểm thử các giao thức đa dạng ngoài HTTP (như JDBC, FTP). Ngược lại, Locust là lựa chọn vượt trội cho các đội ngũ phát triển theo định hướng DevOps, ưu tiên tự động hóa, quản lý phiên bản kịch bản kiểm thử và yêu cầu hiệu năng cao.

extbfĐặc điểm	**Locust**	**K6**
Ngôn ngữ chính	Python	JavaScript (trong Go runtime)
Mô hình cốt lõi	Tests-as-Code	Tests-as-Code
Mô hình đồng thời	Dựa trên sự kiện (Greenlets)	Dựa trên sự kiện (Goroutines)
Giao diện chính	Web UI & Dòng lệnh (CLI)	Ưu tiên Dòng lệnh (CLI)
Phong cách báo cáo	Web UI thời gian thực	CLI trực tiếp & tích hợp bên thứ ba (Grafana, Datadog)
Điểm mạnh chính	Hệ sinh thái Python mạnh mẽ, kịch bản linh hoạt	Hiệu năng thực thi cao, tích hợp sâu với hệ sinh thái Grafana

Nhận xét:
Cả Locust và K6 đều có kiến trúc dựa trên sự kiện hiệu quả, giúp chúng vượt trội hơn JMeter về mặt sử dụng tài nguyên. Sự lựa chọn giữa hai công cụ này thường phụ thuộc vào kỹ năng và hệ sinh thái của đội ngũ phát triển. K6 là lựa chọn tốt cho các nhóm làm việc chủ yếu với JavaScript/TypeScript và đã đầu tư vào ngăn xếp giám sát của Grafana. Trong khi đó, Locust hấp dẫn hơn đối với các đội ngũ backend Python, kỹ sư SRE, hoặc khi kịch bản kiểm thử đòi hỏi logic phức tạp cần đến sức mạnh của các thư viện Python phong phú.

2.3 Locust vs. Gatling

Gatling là một công cụ hiệu năng cao, được xây dựng trên nền tảng JVM (sử dụng Scala) và cũng áp dụng kiến trúc bất đồng bộ dựa trên sự kiện. Giống như Locust và K6, Gatling là một công cụ "tests-as-code" nhưng sử dụng một Ngôn ngữ Đặc tả Miền (DSL) riêng. So sánh Locust và Gatling cho thấy sự đánh đổi giữa hiệu năng thô của JVM và tính dễ sử dụng, linh hoạt của Python.

extbfBảng 4.3: So sánh Locust và Gatling

extbfĐặc điểm	**Locust**	**Gatling**
Ngôn ngữ chính	Python	Scala/Java/Kotlin
Mô hình cốt lõi	Tests-as-Code	Tests-as-Code (thông qua DSL)
Mô hình đồng thời	Dựa trên sự kiện (Greenlets)	Bất đồng bộ dựa trên Actor (Akka)
Giao diện chính	Web UI & Dòng lệnh (CLI)	Giao diện ghi kịch bản (Recorder) & CLI
Phong cách báo cáo	Web UI thời gian thực	Báo cáo HTML chi tiết, đẹp mắt sau kiểm thử
Điểm mạnh chính	Dễ học, linh hoạt, cộng đồng Python lớn	Hiệu năng rất cao trên JVM, báo cáo chi tiết và trực quan

Nhận xét:
Gatling thường được công nhận về hiệu năng thực thi thô và khả năng tạo ra các báo cáo HTML tĩnh rất chi tiết và đẹp mắt. Tuy nhiên, việc sử dụng Scala và DSL của nó có thể tạo ra một rào cản học tập cho các đội ngũ không quen thuộc với hệ sinh thái JVM. Locust, với Python, cung cấp một con đường dễ tiếp cận hơn