

# Báo cáo kiểm thử v1.0

## I. Kịch bản test

Gửi hàng loạt  $x$  yêu cầu POST đến endpoint /api/save (dùng để tạo và lưu nội dung text) trong  $y$  giây, với các test case như sau:

**Bảng 1. Các testcase cho version 1.0**

Testcase	Sample	Loop	Ramp-up period(s)
1	1000	1	5
2	2000	1	5
3	5000	1	5

Trong đó,  $x$  là số sample và  $y$  là ramp-up period(s).

## II. Kết quả test

Sau khi run các testcase trên bằng công cụ Jmeter, em có bảng kết quả test như sau:

**Bảng 2. Các testcase cho version 1.0**

Testcase	Sample	Loop	Ramp-up period(s)	Err (%)	Avg (ms)	TotalTime(s)	Req/s
1	1000	1	5	0	7	5	200
2	2000	1	5	0	18	6	333
3	5000	1	5	0	41	5	1000
4	10000	1	5	3.01	137	10	1000

Trong đó:

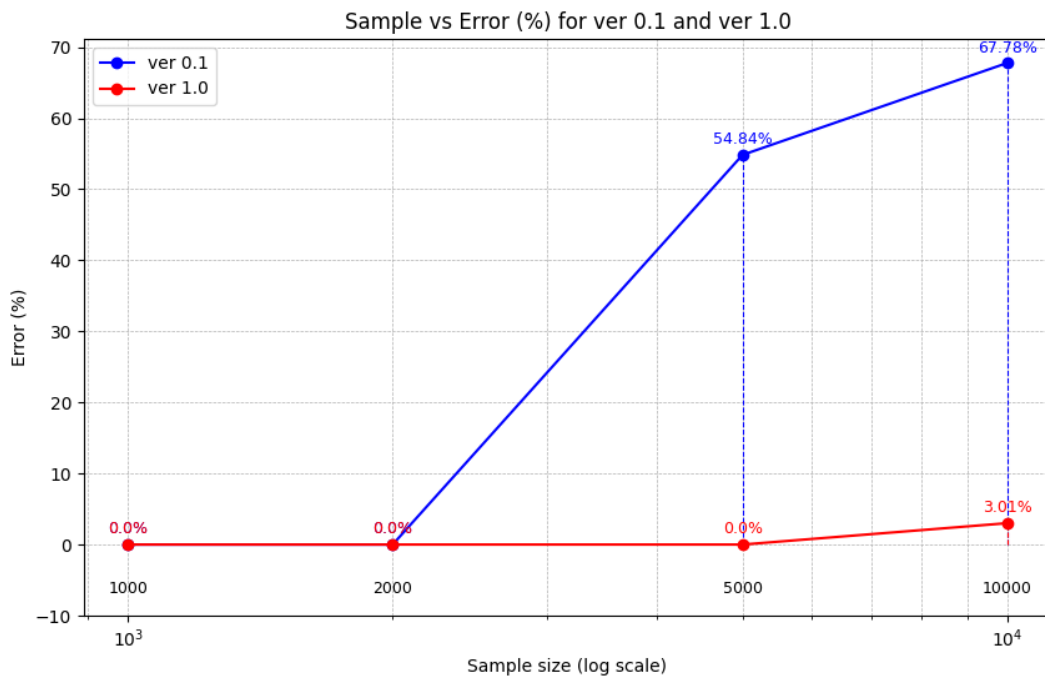
Tham số	Ý nghĩa
Sample	Tổng số yêu cầu đến hệ thống. (bằng số luồng)
Loop	Số lần lặp lại test, để lấy trung bình kết quả
Ramp-up period	Thời gian khởi động tất cả các luồng (thread) đã được cấu hình.
Err	Tỉ lệ request không thành công
Avg	Thời gian phản hồi trung bình (Response Time) của request.
TotalTime	Tổng thời gian chạy Test
Req/s	Số request trung bình gửi mỗi giây trong suốt bài test

### III. So sánh giữa version 1.0 và version 0.1

Dưới đây là biểu đồ và bảng so sánh tỷ lệ lỗi giữa version 1.0 đã áp dụng Queue-Based Load Leveling pattern cho POST query và version 0.1 (phiên bản đầu tiên) không có queue message.

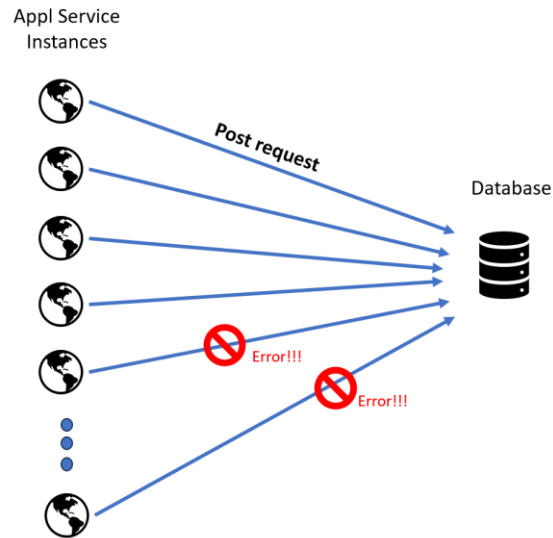
**Bảng 3. Biểu đồ so sánh tỷ lệ lỗi giữa version 0.1 và version 1.0**

Testcase	Sample	Loop	Ramp-up period(s)	Err (%)	
				Version 0.1	Version 1.0
1	1000	1	5	0	0
2	2000	1	5	0	0
3	5000	1	5	54,84	0
4	10000	1	5	67,78	3.01



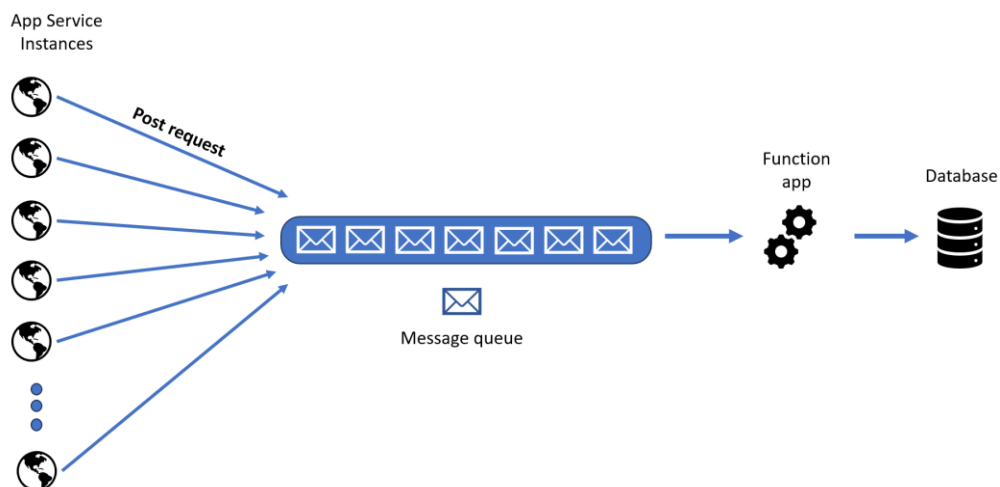
**Hình 1. Biểu đồ so sánh tỷ lệ lỗi giữa version 0.1 và version 1.0**

Sau khi áp dụng message queue cho POST query trong version 1.0, tỷ lệ lỗi đã giảm đáng kể từ 54.84% xuống 0.00% với số request là 5000 và từ 67.78% xuống 3.01% với số request là 10000. Do POST query là việc người dùng tạo ra 1 pastebin và ghi vào database nên trong version 0.1, việc trong cùng một thời điểm có quá nhiều request yêu cầu ghi vào database (tác vụ tốn thời gian) khiến cho việc xử lý bị quá tải dẫn đến các truy vấn không thành công (minh họa trong hình 2).



**Hình 2. Minh họa quá trình thực hiện POST query trong version 0.1**

Trong version 1.0, việc triển khai message queue theo Queue-Based Load Leveling pattern đã tách biệt việc tiếp nhận các request là POST request và việc xử lý (ghi vào database). Khi đó, tốc độ tiếp nhận các yêu cầu không cần phải tương quan với tốc độ xử lý các yêu cầu đó và các yêu cầu sẽ lần lượt được lấy ra từ queue và xử lý (minh họa trong hình 3). Do đó, pattern này đồng thời giúp independent scale tốc độ xử lý các yêu cầu với tốc độ tiếp nhận các yêu cầu, thuận tiện cho việc sử dụng Competing Consumers pattern trong version tiếp theo để tăng tốc độ xử lý các yêu cầu ở trong queue.



**Hình 3. Minh họa quá trình thực hiện POST request trong version 1.0**