

Báo cáo cuối kỳ
Seminar các vấn đề hiện đại của CNPM

SO SÁNH HIỆU SUẤT

Go vs Java

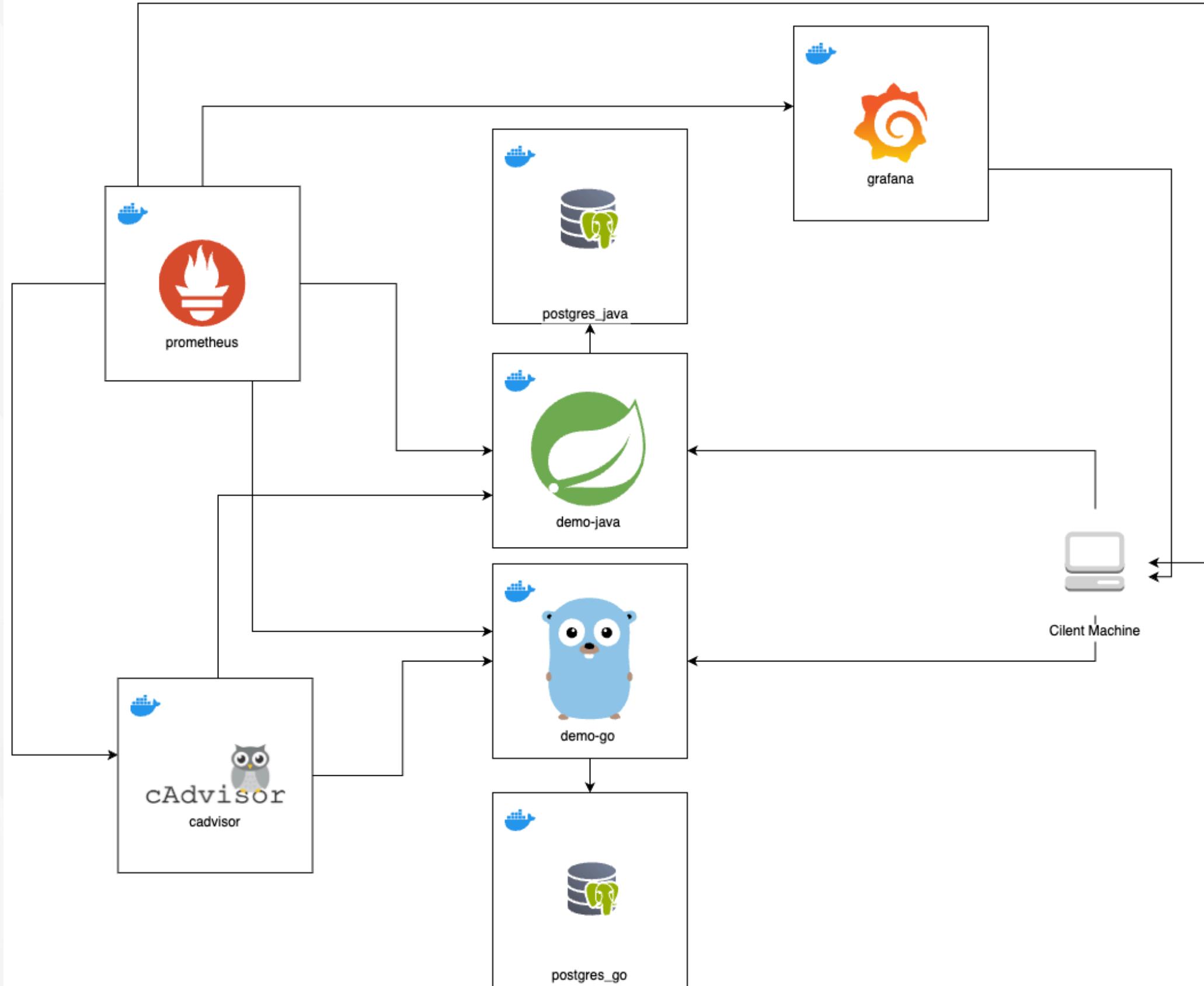
GVHD:

Ths. Đinh Nguyễn Anh Dũng

Sinh viên:

- Nguyễn Đình Khoa - 21520997
- Nguyễn Tiến Vĩ - 21522788

1. Thiết lập hệ thống



demo-java

Container cho server Spring Boot

Cấu hình: 2 CPU, 2GB RAM

demo-go

Container cho server Gin

Cấu hình: 2 CPU, 2GB RAM

prometheus

Container cho service Prometheus, có nhiệm vụ thu thập dữ liệu tài nguyên, hiệu suất

cadvisor

Container cho service cAdvisor: theo dõi mức sử dụng tài nguyên của các Docker Container

grafana

Container cho service Grafana, có nhiệm vụ trực quan hóa dữ liệu từ prometheus

Client Machine

Gửi yêu cầu đến server

2. Phương pháp thực hiện

Load test để có được các thông số:

- CPU
- RAM
- Latency

Thực hiện các trường hợp test khác nhau:

Trạng thái idle

Get static json

Create 1 order

API thực hiện nhiều thao tác với DB: kiểm tra stock, update stock, tính tổng tiền, tạo order.

Get 1 order

API thực hiện việc truy vấn database lấy 1 record.

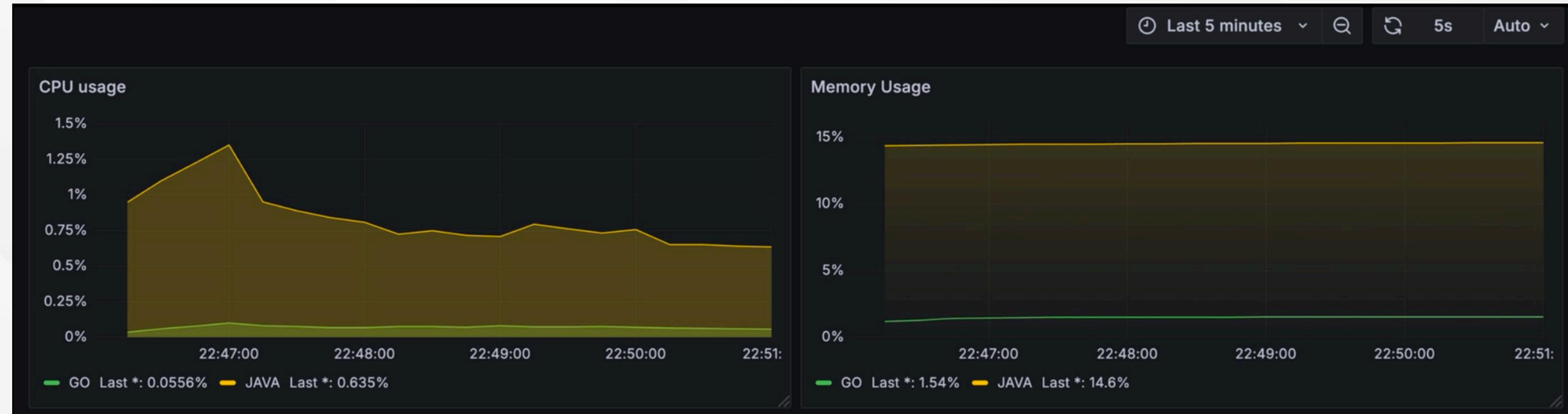
Get all products

API thực hiện việc truy vấn database lấy nhiều record.

Convert ảnh màu sang monochrome

3. Kết quả thực hiện

So sánh trạng thái Idle



Gin

CPU

Ổn định trong khoảng 0.1%

RAM

Ổn định ở mức 1.5%

Spring Boot

Lúc đầu sử dụng nhiều CPU và tăng dần, sau khoảng 1 phút khởi động thì giảm dần và dao động từ 0.7% - 0.8%

Ổn định ở mức 14.5%

So sánh GET Static JSON



	Gin	Spring Boot
CPU	Sử dụng CPU tăng dần theo số lượng request, đạt đỉnh ở khoảng 12% tại mức 800 requests/s.	Mức sử dụng CPU cao, tăng dần lên đến 20% vào lúc số lượng request chỉ khoảng 170 requests/s. Tuy nhiên, sau đó giảm dần và dao động trong khoảng 15%.
RAM	Ổn định ở mức 0.8% trong suốt quá trình.	Sử dụng khoảng 13.5% RAM lúc idle, khi số lượng request tăng cao thì tăng đến khoảng 14.6% và giữ ổn định
Latency	p99 Latency ở những request ban đầu cao (9ms), sau đó giảm dần và ổn định ở mức 3.7ms	p99 Latency ban đầu rất cao (45ms), sau đó giảm dần và ổn định tại mức 4.7ms

So sánh POST Create Order



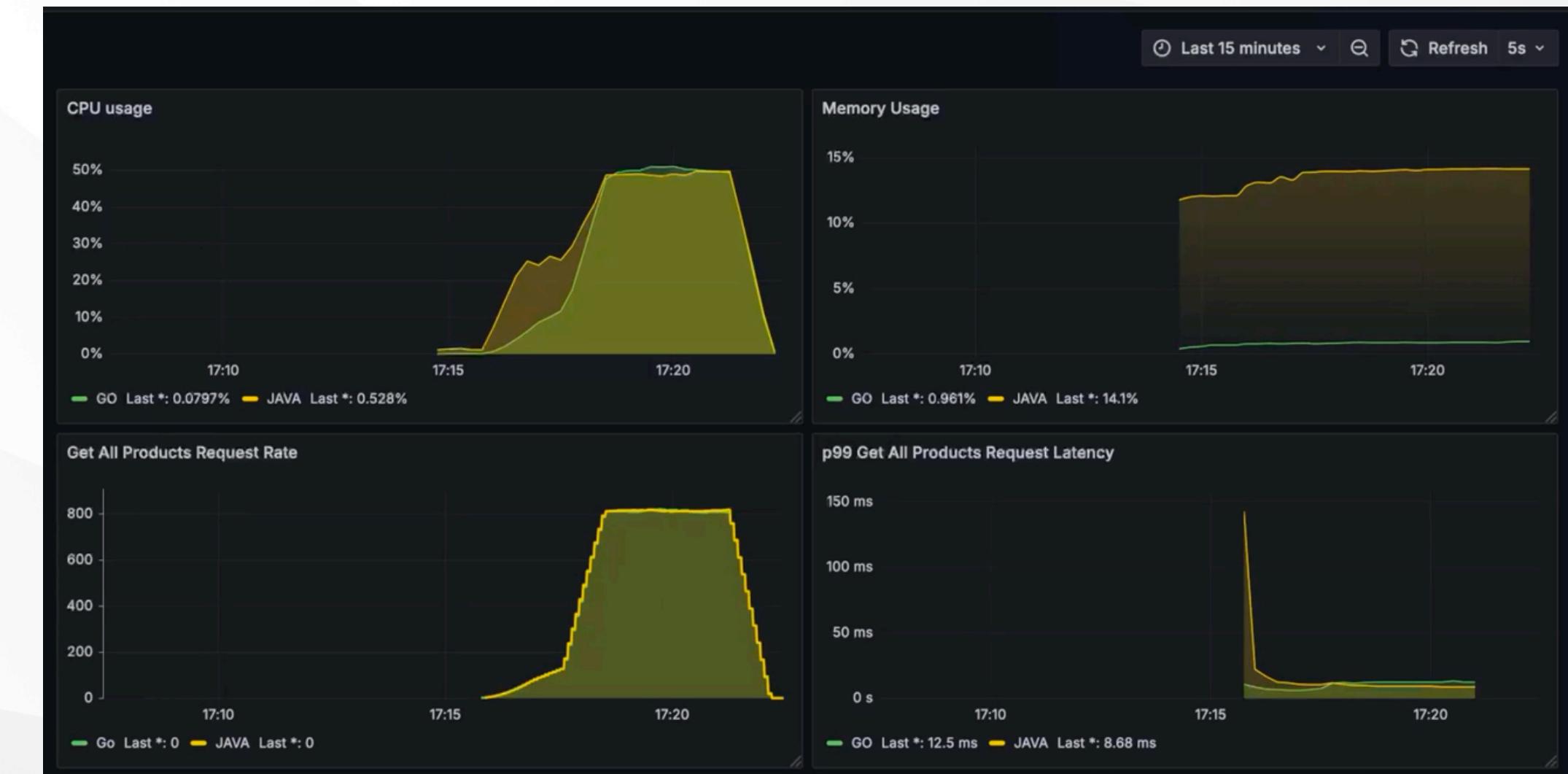
	Gin	Spring Boot
CPU	Mức sử dụng CPU tăng rất nhanh so với các bài test khác, lên đến khoảng 160% tại thời điểm database bị quá tải.	Mức sử dụng CPU cao, lên đến khoảng 50% tại 620 requests/s.
RAM	Ban đầu, mức sử dụng RAM thấp (0.8%). Tuy nhiên, khi database bắt đầu có hiện tượng quá tải thì tăng đột ngột lên 21%	Sử dụng khoảng 12.5% RAM lúc idle, tăng khi bắt đầu nhận được requests, lên đến khoảng 14.5%. Khi database bị quá tải thì RAM tăng đột ngột lên khoảng 19%
Latency	p99 Latency ban đầu ổn định ở mức 10ms. Khi database bị quá tải thì khả năng đáp ứng giảm rõ rệt, p99 Latency tăng lên đến khoảng 2.6s	p99 Latency ban đầu cao (110ms), sau đó ổn định và giảm xuống mức 10ms. Khi database bị quá tải thì khả năng đáp ứng cũng giảm rõ rệt, p99 latency tăng lên 1.44s

So sánh GET 1 Order



	Gin	Spring Boot
CPU	Sử dụng CPU tăng dần theo số lượng request, đạt đỉnh ở khoảng 18% tại mức 850 requests/s.	Mức sử dụng CPU cao, tăng dần lên đến 20% vào lúc số lượng request chỉ khoảng 170 requests/s. Tuy nhiên, sau đó giảm dần và dao động trong khoảng 15%.
RAM	Dao động trong khoảng từ 0.85-1.4%. Mức sử dụng RAM tăng khi số requests tăng cao.	Sử dụng khoảng 12.5% RAM lúc idle, tăng khi bắt đầu nhận được requests, lên đến khoảng 14.5% ở mức 850 requests/s
Latency	p99 Latency ở những request ban đầu ổn định ở mức khoảng 6.5ms. Tuy nhiên, về giai đoạn cuối bài test, khi số lượng request tăng nhiều thì latency cũng tăng đến khoảng 10.5ms	p99 Latency ban đầu rất cao (119ms), sau đó giảm dần và ổn định tại mức 7.5ms-8.5ms

So sánh GET All Products



	Gin	Spring Boot
CPU	Lúc bắt đầu tăng vừa phải, khi số lượng request nhiều thì tăng nhanh và vượt qua cả Spring Boot tại 800 req/s với khoảng 50%	Lúc bắt đầu tăng nhanh, đến khoảng 25% thì có khoảng dừng sau đó lại tăng nhanh, đạt mức 49% tại 800 request/s
RAM	Ổn định ở mức 0.8% - 0.9% trong suốt quá trình.	Tăng nhẹ từ mức 12% lúc idle đến khoảng 14% khi số lượng request tăng cao và giữ ổn định.
Latency	p99 Latency dao động trong khoảng từ 9ms-12.5ms	p99 Latency ban đầu rất cao (143ms), sau đó giảm dần và dao động trong khoảng 8ms-9ms

So sánh Convert Image to Monochrome



	Gin	Spring Boot
CPU	Mức sử dụng CPU tăng nhanh từ lúc nhận request, đến mức 110% tại 125 requests/s.	Mức sử dụng CPU tăng rất nhanh, đạt mức 200% (sử dụng hết 2CPU) tại 125 requests/s.
RAM	Ổn định trong khoảng dưới 1%	Tăng nhanh từ 12% đến khoảng 34% khi nhận được requests
Latency	p99 Latency dao động trong khoảng từ 25ms-32ms	p99 Latency ban đầu thấp (tương đương với Gin), tuy nhiên, khi số lượng request tăng cao thì tốc độ xử lý giảm, độ trễ tăng lên đến 2s, sau đó giảm xuống mức 1.9s.

4. Tóm tắt kết quả

Tóm tắt

CPU

Gin có xu hướng sử dụng CPU hiệu quả hơn trong các trường hợp tải thấp và trung bình. Tuy nhiên khi tải cao (800 requests/s), Gin tiêu tốn tương đương hoặc thậm chí cao hơn Spring Boot. Trong các tác vụ phức tạp như chuyển đổi ảnh, Gin sử dụng ít CPU hơn đáng kể so với Spring Boot.

Memory

Memory: Gin ổn định với mức sử dụng RAM rất thấp (1-1.5%) so với Spring Boot (12%-34%) trong các bài test, cho thấy lợi thế của Gin trong việc quản lý bộ nhớ.

Latency

Gin có độ trễ thấp hơn Spring Boot trong hầu hết các tình huống, đặc biệt với tốc độ xử lý nhanh hơn đáng kể trong tác vụ phức tạp như chuyển đổi ảnh (25ms so với 1.9s).

Với tác vụ đọc dữ liệu dữ liệu JSON tĩnh hay từ cơ sở dữ liệu, cả hai Framework đều ổn định với độ trễ thấp sau khi hệ thống đạt trạng thái cân bằng.

Database

Gin tỏ ra hiệu quả hơn Spring Boot trong bài test đọc dữ liệu từ database. Tuy nhiên, ở bài test ghi dữ liệu, Gin có kết quả kém hơn Java rõ rệt. Điều này cho thấy Spring Boot có độ tin cậy cao hơn khi tương tác với database.

Thank You

