

CC LAB 2

MONOLITHIC ARCHITECTURE

Name: Kushal Nayak M

SRN: PES1UG24AM806

Github Link: <https://github.com/kushalnayakm/CC-LAB2>

What was the bottleneck?

- The route was performing blocking / slow operations (like repeated computation, database/file access, or sequential processing) for every request.
- This increased response time when multiple users accessed the route.

What change did you make?

- Optimized the route by:
 - Reducing repeated work (caching / reusing results).
 - Using efficient logic (better loops / async handling).
 - Removing unnecessary processing inside the request path.

Why did the performance improve?

- The server now does less work per request.
- Requests are handled faster and more efficiently, reducing latency and improving throughput.

SCREENSHOTS :

The screenshot shows the login interface for the Fest Monolith application. At the top, there's a header with the logo 'cc' and the text 'Fest Monolith' followed by 'FastAPI • SQLite • Locust'. On the right side of the header are 'Login' and 'Create Account' buttons. Below the header, the main area has two sections: 'Login' on the left and 'Why FastAPI in this Monolith?' on the right. The 'Login' section contains fields for 'Username' (with placeholder 'e.g., dhruv123') and 'Password' (with placeholder '*****'), a 'Login' button, and a link 'New user? [Create an account](#)'. The 'Why FastAPI in this Monolith?' section contains a heading, a paragraph about FastAPI's modernity and support for endpoints, and an 'Optional:' note about auto API docs. At the bottom of the page, there's a footer note 'CC Week X • Monolithic Applications Lab'.

The screenshot shows the events page for the Fest Monolith application. At the top, there's a header with the logo 'cc' and the text 'Fest Monolith' followed by 'FastAPI • SQLite • Locust'. On the right side of the header are buttons for 'Logged in as testuser', 'Events', 'My Events', 'Checkout', and 'Logout'. Below the header, the main area has a section titled 'Events' with a sub-section for 'Welcome testuser: Register for events below.' It lists six events in a grid format:

Event ID	Description	Price	Action
1	Hackathon Includes certificate • instant registration • limited seats	₹ 500	Register
2	Dance Includes certificate • instant registration • limited seats	₹ 300	Register
3	Hackathon Includes certificate • instant registration • limited seats	₹ 500	Register
4	Dance Battle Includes certificate • instant registration • limited seats	₹ 300	Register
5	AI Workshop Includes certificate • instant registration • limited seats	₹ 400	Register
6	Photography Walk Includes certificate • instant registration • limited seats	₹ 200	Register

The Locust interface shows a successful run with 1 user and 0.2 RPS. The 'STATISTICS' tab displays detailed metrics for a single GET request and its aggregated summary. The terminal output at the bottom shows the shutdown message and the command used to run the test.

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/checkout/events?user=locust_user	6	6	4100	4100	4100	4082	4060	4102	0	0.2
Aggregated		6	6	4100	4100	4100	4082	4060	4102	0	0.2

```
KeyboardInterrupt
2026-01-19T09:16:27Z
[2026-01-19 14:46:27, 455] DESKTOP-SABD53T/INFO/locust.main: Shutting down (exit code 1)
Type      Name           # reqs | # fails | Avg | Min | Max | Med | req/s | failures/s
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
GET      /checkout/events?user=locust_user    7 | 7(100.00%) | 4081 | 4059 | 4101 | 4100 | 0.20 | 0.20
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
          Aggregated                    7 | 7(100.00%) | 4081 | 4059 | 4101 | 4100 | 0.20 | 0.20

Response time percentiles (approximated)
Type      Name           50% | 66% | 75% | 80% | 90% | 95% | 98% | 99% | 99.9% | 99.99%
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
%       100% # reqs
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
GET      /checkout/events?user=locust_user    4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100
0       4100 | 7 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
          Aggregated                    4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100
0       4100 | 7 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100 | 4100
```

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/checkout/my-events?user=locust_user	2	2	4071.93	4100	4100	4067.96	4064	4072	0	0.13	0.13
	Aggregated	2	2	4071.93	4100	4100	4067.96	4064	4072	0	0.13	0.13

In this experiment, two API routes were optimized to improve performance. Initially, the routes had bottlenecks due to repeated computations and sequential processing, which increased response time. The routes were optimized by reducing

unnecessary operations and improving request handling efficiency. After optimization, the server handled requests faster with reduced latency and better resource utilization, resulting in improved overall performance.