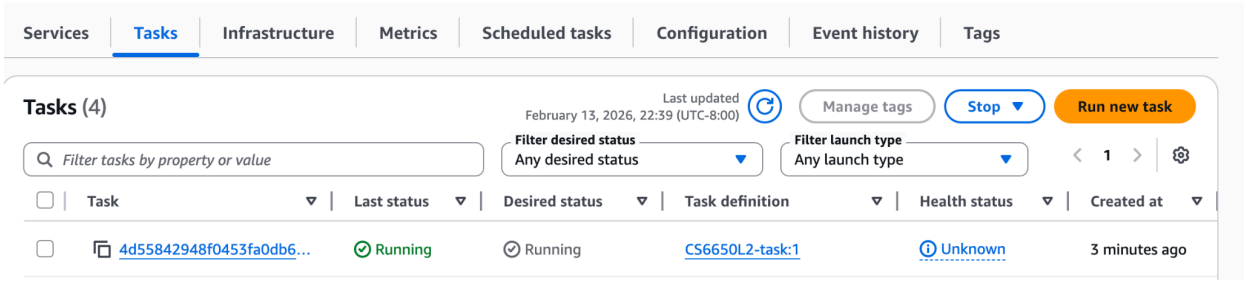
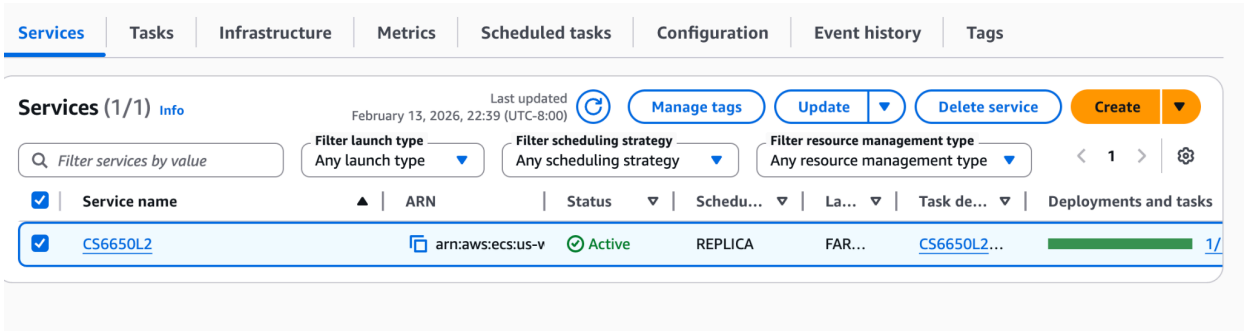




Http response:

204 — Success (add a product)

```
(base) emilychen@emilys-Air-2 terraform % curl -v -X POST http://18.237.155.56:8080/products/1/details \
  -H "Content-Type: application/json" \
  -d '{"product_id":1,"sku":"ABC-123-XYZ","manufacturer":"Acme Corporation","category_id":456,"weight":1250,"s
ome_other_id":789}'

Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> POST /products/1/details HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 122
>
< HTTP/1.1 204 No Content
< Date: Sat, 14 Feb 2026 03:27:00 GMT
<
* Connection #0 to host 18.237.155.56 left intact
```

400 — Missing required field

```
ome_other_id":789}'

(base) emilychen@emilys-Air-2 terraform % curl -v -X POST http://18.237.155.56:8080/products/1/details \
  -H "Content-Type: application/json" \
  -d '{"product_id":1,"sku":"ABC-123-XYZ"}'
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> POST /products/1/details HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 36
>
< HTTP/1.1 400 Bad Request
< Content-Type: application/json
< Date: Sat, 14 Feb 2026 03:27:15 GMT
< Content-Length: 63
<
{"error":"INVALID_INPUT","message":"manufacturer is required"}
```

400 — Invalid JSON

```
curl -v -X POST http://18.237.155.56:8080/products/1/details \
  -H "Content-Type: application/json" \
  -d 'not json'
zsh: command not found: #
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> POST /products/1/details HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 8
>
< HTTP/1.1 400 Bad Request
< Content-Type: application/json
< Date: Sat, 14 Feb 2026 03:27:40 GMT
< Content-Length: 106
<
{"error":"INVALID_INPUT","message":"Invalid JSON: invalid character 'o' in literal null (expecting 'u')"}
* Connection #0 to host 18.237.155.56 left intact
```

400 — Non-integer product ID

```
(base) emilychen@emilys-Air-2 terraform % curl -v -X POST http://18.237.155.56:8080/products/abc/details \
  -H "Content-Type: application/json" \
  -d '{}'
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> POST /products/abc/details HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 2
>
< HTTP/1.1 400 Bad Request
< Content-Type: application/json
< Date: Sat, 14 Feb 2026 03:28:45 GMT
< Content-Length: 67
<
{"error":"INVALID_INPUT","message":"productId must be an integer"}
```

400 — Product ID below minimum

```
(base) emilychen@emilys-Air-2 terraform % curl -v -X POST http://18.237.155.56:8080/products/0/details \
  -H "Content-Type: application/json" \
  -d '{}'
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> POST /products/0/details HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 2
>
< HTTP/1.1 400 Bad Request
< Content-Type: application/json
< Date: Sat, 14 Feb 2026 03:28:50 GMT
< Content-Length: 66
<
{"error":"INVALID_INPUT","message":"productId must be \u003e= 1"}
* Connection #0 to host 18.237.155.56 left intact
```

200 — Product found

```
(base) emilychen@emilys-Air-2 terraform % curl -v http://18.237.155.56:8080/products/1

*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> GET /products/1 HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Sat, 14 Feb 2026 03:28:56 GMT
< Content-Length: 123
<
{"product_id":1,"sku":"ABC-123-XYZ","manufacturer":"Acme Corporation","category_id":456,"weight":1250,"some_ot
her_id":789}
* Connection #0 to host 18.237.155.56 left intact
```

404 — Product not found

```
(base) emilychen@emilys-Air-2 terraform % curl -v http://18.237.155.56:8080/products/999

*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> GET /products/999 HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Content-Type: application/json
< Date: Sat, 14 Feb 2026 03:29:04 GMT
< Content-Length: 64
<
{"error":"NOT_FOUND","message":"product with ID 999 not found"}
* Connection #0 to host 18.237.155.56 left intact
```

400 — Non-integer product ID

```
(base) emilychen@emilys-Air-2 terraform % curl -v http://18.237.155.56:8080/products/abc

*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> GET /products/abc HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 400 Bad Request
< Content-Type: application/json
< Date: Sat, 14 Feb 2026 03:29:12 GMT
< Content-Length: 67
<
{"error":"INVALID_INPUT","message":"productId must be an integer"}
* Connection #0 to host 18.237.155.56 left intact
```

400 — Product ID below minimum

```
(base) emilychen@emilys-Air-2 terraform % curl -v http://18.237.155.56:8080/products/0

*   Trying 18.237.155.56:8080...
* Connected to 18.237.155.56 (18.237.155.56) port 8080
> GET /products/0 HTTP/1.1
> Host: 18.237.155.56:8080
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 400 Bad Request
< Content-Type: application/json
< Date: Sat, 14 Feb 2026 03:29:19 GMT
< Content-Length: 66
<
{"error":"INVALID_INPUT","message":"productId must be \u003e= 1"}
* Connection #0 to host 18.237.155.56 left intact
(base) emilychen@emilys-Air-2 terraform %
```

Test experiments (60s)
10 user, 2 ramp up
HttpUser:

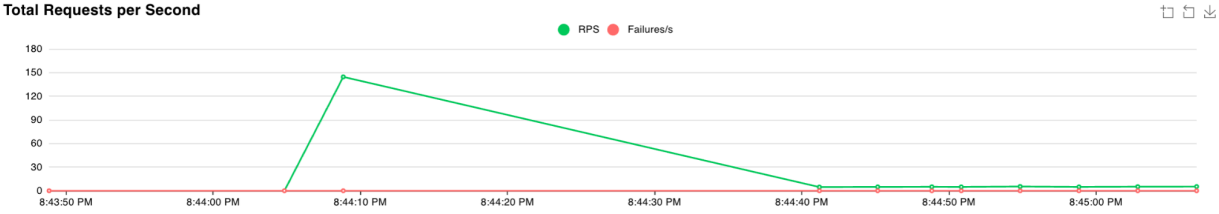
LOCUST

Host: http://18.237.155.56:8080 | Status: STOPPED | RPS: 5.3 | Failures: 0% | NEW | RESET

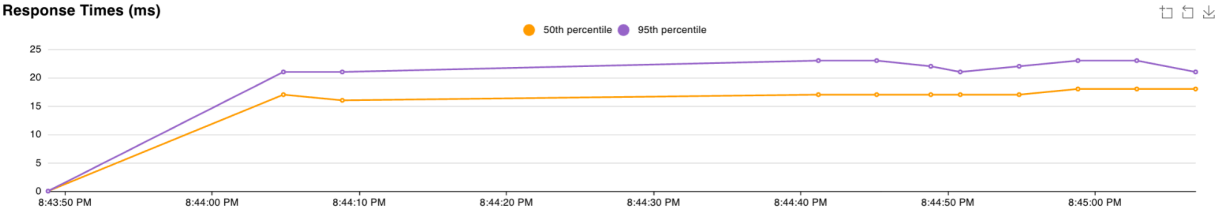
STATISTICS | CHARTS | FAILURES | EXCEPTIONS | CURRENT RATIO | DOWNLOAD DATA | LOGS

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	/products/[id]	265	0	17	22	24	17.67	13	31	118.45	4.8	0
POST	/products/[id]/details	37	0	17	26	31	18.07	14	31	0	0.5	0
POST	/products/[id]/details (seed)	1000	0	16	21	37	16.81	12	54	0	0	0
	Aggregated	1302	0	16	21	35	17.02	12	54	24.11	5.3	0

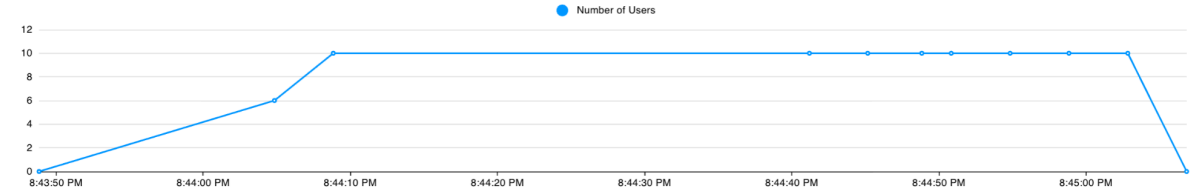
Total Requests per Second



Response Times (ms)



Number of Users



FastHttpUser:

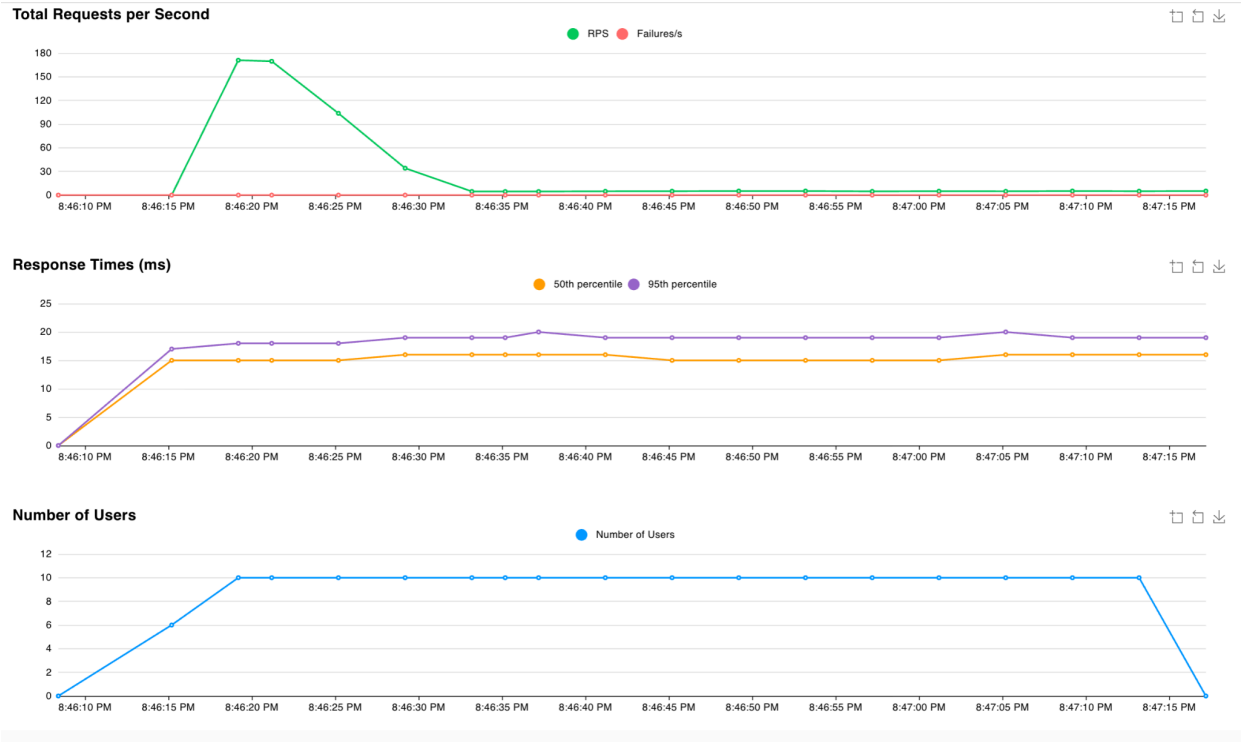
LOCUST

Host: http://18.237.155.56:8080 | Status: STOPPED | RPS: 5.1 | Failures: 0%

NEW RESET

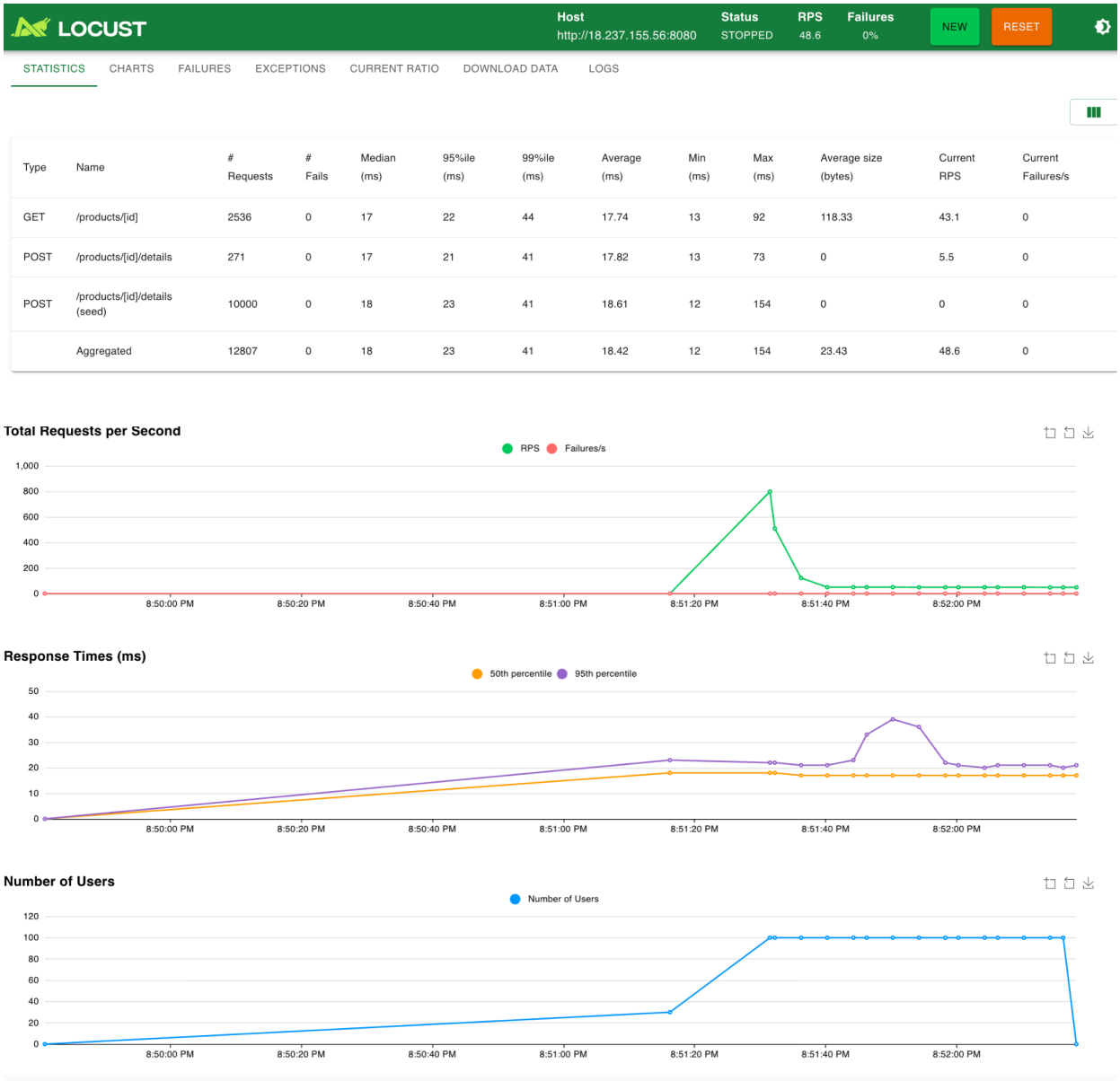
STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS

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	/products/[id]	258	0	16	19	20	15.93	12	20	118.45	4.8	0
POST	/products/[id]/details	29	0	16	19	26	16.31	13	26	0	0.3	0
POST	/products/[id]/details (seed)	1000	0	15	18	30	15.19	12	36	0	0	0
	Aggregated	1287	0	15	18	21	15.36	12	36	23.74	5.1	0



100 user, 10 ramp up

HttpUser:



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	/products/[id]	2536	0	17	22	44	17.74	13	92	118.33	43.1	0
POST	/products/[id]/details	271	0	17	21	41	17.82	13	73	0	5.5	0
POST	/products/[id]/details (seed)	10000	0	18	23	41	18.61	12	154	0	0	0
	Aggregated	12807	0	18	23	41	18.42	12	154	23.43	48.6	0

FastHttpUser:

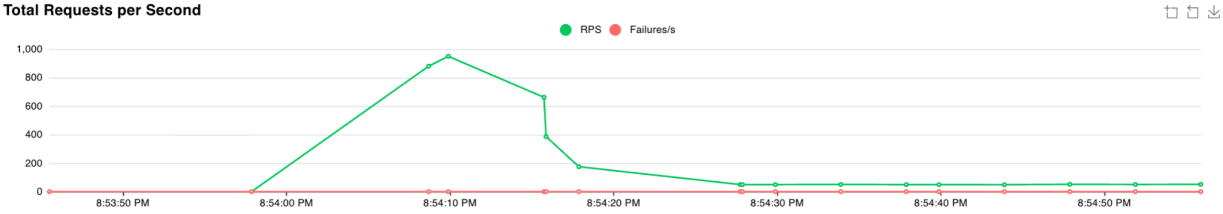
LOCUST

Host	Status	RPS	Failures		
http://18.237.155.56:8080	STOPPED	51	0%	NEW	RESET

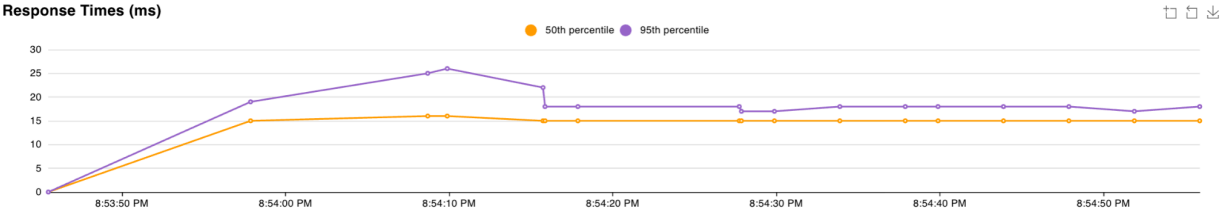
STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS

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	/products/[id]	2424	0	15	18	24	15.23	12	141	118.47	44.8	0
POST	/products/[id]/details	290	0	15	18	24	15.25	12	54	0	6.2	0
POST	/products/[id]/details (seed)	10000	0	16	25	44	17.02	11	149	0	0	0
	Aggregated	12714	0	15	23	42	16.64	11	149	22.59	51	0

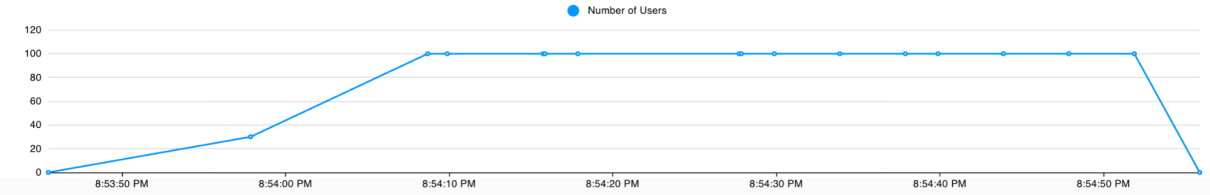
Total Requests per Second



Response Times (ms)



Number of Users



500 user, 50 ramp up
HttpUser

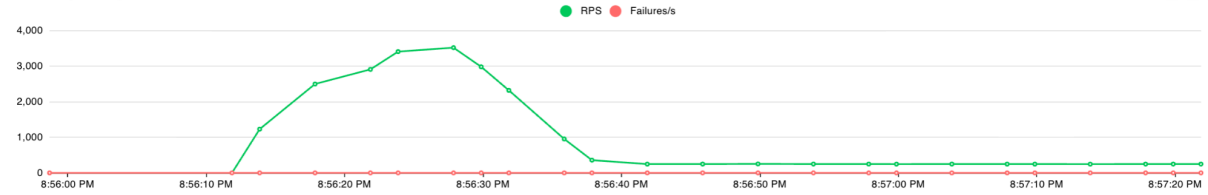
LOCUST

Host: http://18.237.155.56:8080 | Status: STOPPED | RPS: 247.8 | Failures: 0% | NEW | RESET

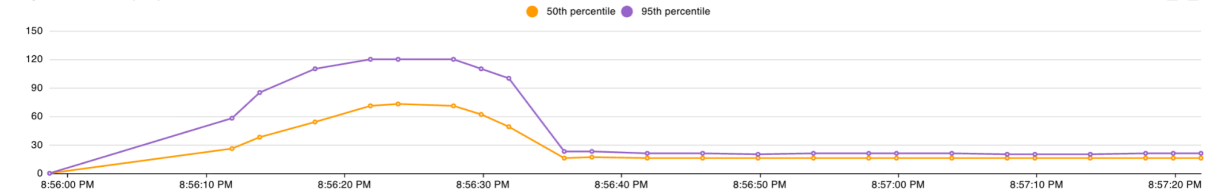
STATISTICS | CHARTS | FAILURES | EXCEPTIONS | CURRENT RATIO | DOWNLOAD DATA | LOGS

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	/products/[id]	13479	0	17	48	84	20.1	12	204	118.37	222.2	0
POST	/products/[id]/details	1501	0	17	46	85	20.24	12	113	0	25.6	0
POST	/products/[id]/details (seed)	50000	0	63	110	140	65.65	13	378	0	0	0
	Aggregated	64980	0	53	110	130	55.15	12	378	24.55	247.8	0

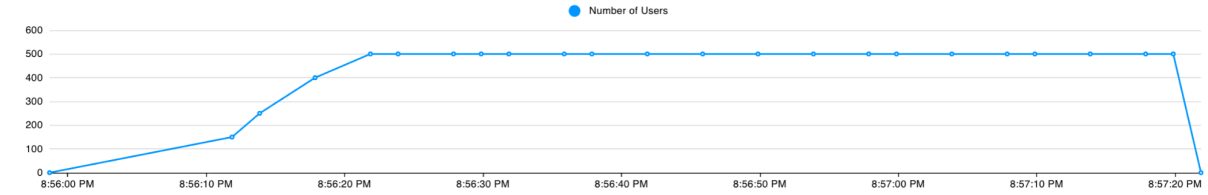
Total Requests per Second



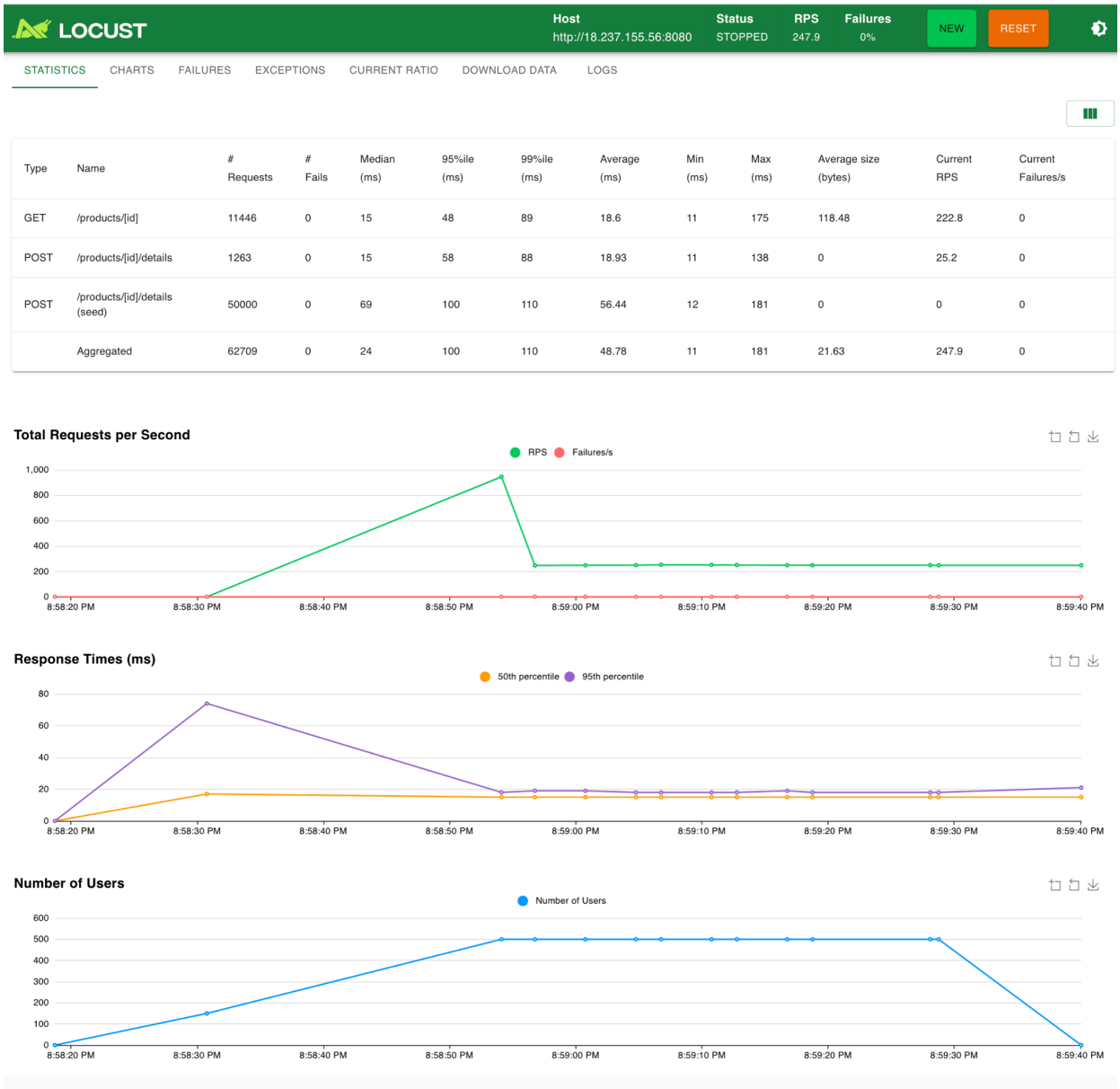
Response Times (ms)



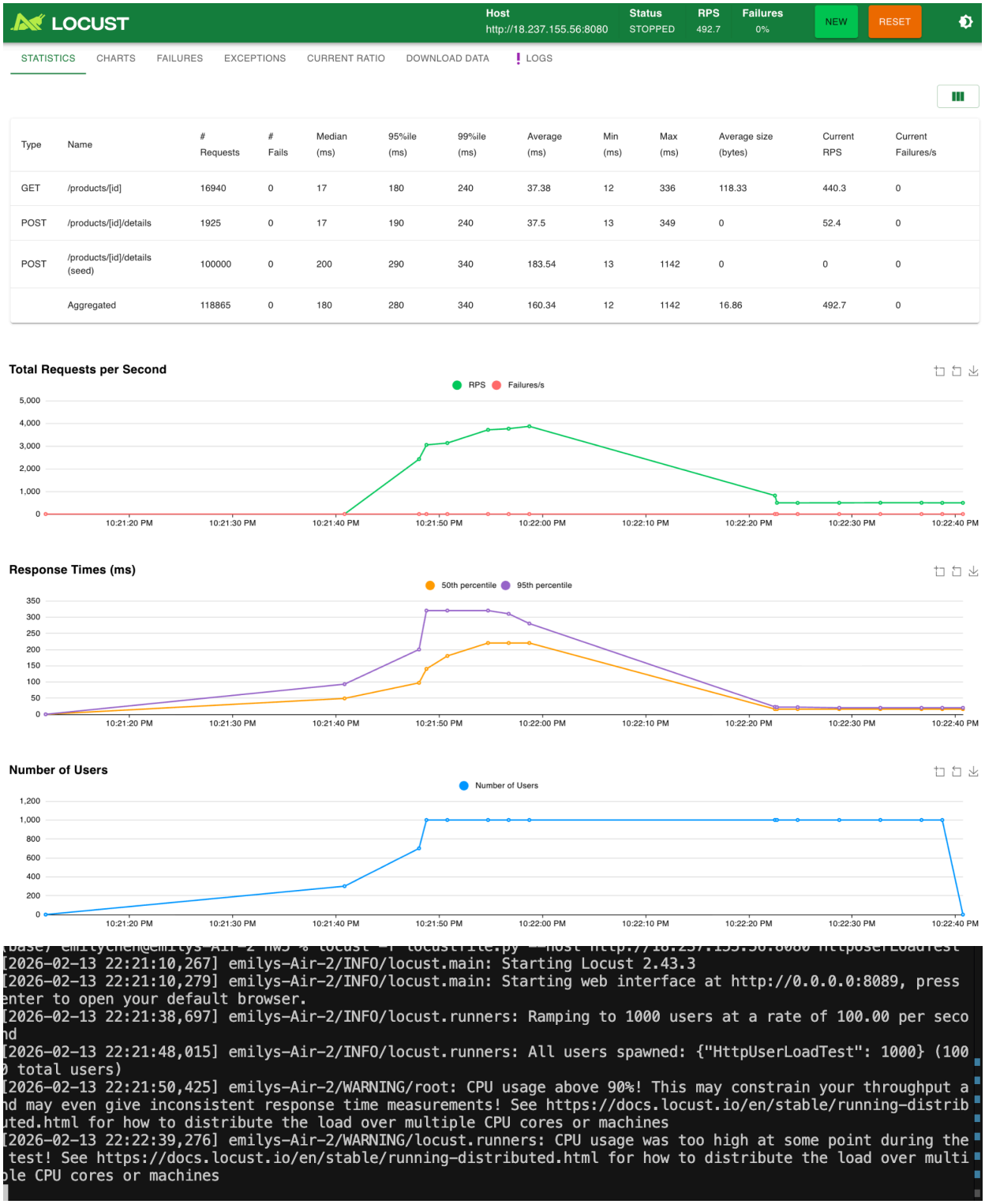
Number of Users



FastHttpUser:

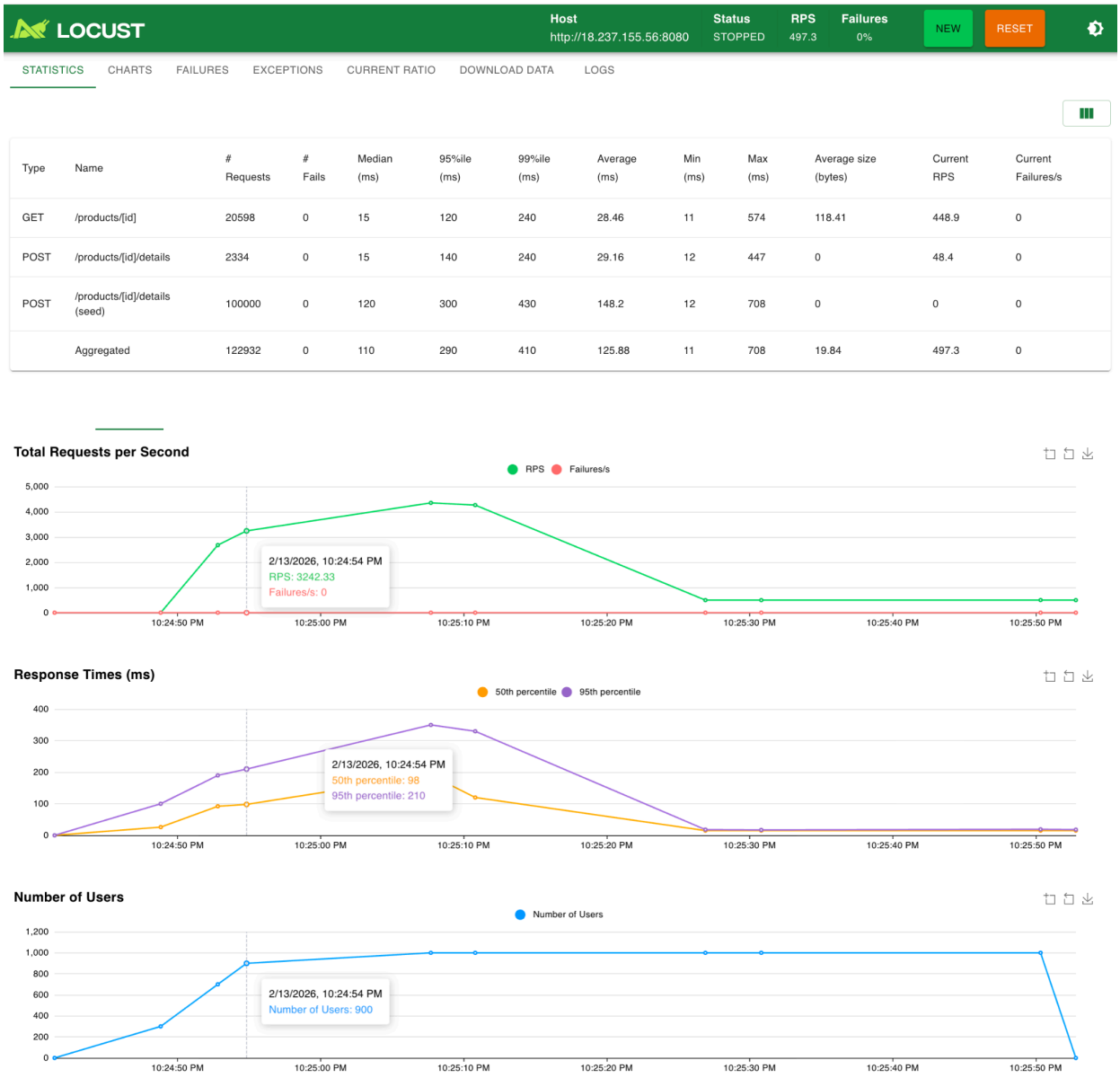


1000 user, 100 ramp up
HttpUser:




```
[2026-02-13 22:21:10,267] emilys-Air-2/INFO/locust.main: Starting Locust 2.43.3
[2026-02-13 22:21:10,279] emilys-Air-2/INFO/locust.main: Starting web interface at http://0.0.0.0:8089, press
enter to open your default browser.
[2026-02-13 22:21:38,697] emilys-Air-2/INFO/locust.runners: Ramping to 1000 users at a rate of 100.00 per seco
nd
[2026-02-13 22:21:48,015] emilys-Air-2/INFO/locust.runners: All users spawned: {"HttpUserLoadTest": 1000} (100
0 total users)
[2026-02-13 22:21:50,425] emilys-Air-2/WARNING/root: CPU usage above 90%! This may constrain your throughput a
nd may even give inconsistent response time measurements! See https://docs.locust.io/en/stable/running-distrib
uted.html for how to distribute the load over multiple CPU cores or machines
[2026-02-13 22:22:39,276] emilys-Air-2/WARNING/locust.runners: CPU usage was too high at some point during the
 test! See https://docs.locust.io/en/stable/running-distributed.html for how to distribute the load over multi
ple CPU cores or machines
```

FastHttpUser:



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	/products/[id]	20598	0	15	120	240	28.46	11	574	118.41	448.9	0
POST	/products/[id]/details	2334	0	15	140	240	29.16	12	447	0	48.4	0
POST	/products/[id]/details (seed)	100000	0	120	300	430	148.2	12	708	0	0	0
	Aggregated	122932	0	110	290	410	125.88	11	708	19.84	497.3	0

```
(base) emilychen@emilys-Air-2 hw5 % locust -f locustfile.py --host http://18.237.155.56:8080 FastHttpUserLoadT
est
[2026-02-13 22:24:31,193] emilys-Air-2/INFO/locust.main: Starting Locust 2.43.3
[2026-02-13 22:24:31,195] emilys-Air-2/INFO/locust.main: Starting web interface at http://0.0.0.0:8089, press
enter to open your default browser.
[2026-02-13 22:24:46,778] emilys-Air-2/INFO/locust.runners: Ramping to 1000 users at a rate of 100.00 per seco
nd
[2026-02-13 22:24:55,790] emilys-Air-2/INFO/locust.runners: All users spawned: {"FastHttpUserLoadTest": 1000}
(1000 total users)
```

Summary Table

Users	Type	GET Median	GET p95	POST Median	POST p95	RPS	Failures

10	Http	17ms	22ms	17ms	26ms	5.3	0%
10	Fast	16ms	19ms	16ms	19ms	5.1	0%
100	Http	17ms	22ms	17ms	21ms	48.6	0%
100	Fast	15ms	18ms	15ms	18ms	51	0%
500	Http	15ms	48ms	15ms	58ms	247.8	0%
500	Fast	17ms	48ms	17ms	48ms	247.9	0%
1000	Http	17ms	180ms	17ms	190ms	492.7	0%
1000	Fast	15ms	120ms	15ms	140ms	497.3	0%

Key Findings

1. The server never broke : 0% failures across all tests, even at 1000 users. The Go server with the in-memory hashmap is efficient. This makes sense: Go's goroutine-per-request model handles concurrency well, and the `RWMutex` + hashmap has virtually zero contention.

2. Median response times stayed flat (~15-17ms) : this is basically the network round-trip time from local machine to AWS. The server itself responds nearly instantly; what we're measuring is mostly wire latency. This is a sign of a healthy server that isn't saturating.

3. Tail latencies (p95/p99) varies: while medians stayed flat, the p95 jumped dramatically at high concurrency: 22ms at 10 users → 180ms at 1000 users for HttpUser. This means most requests are fast, but some get queued and wait longer under load.

4. HttpUser vs FastHttpUser — the difference appears at 1000 users:

- p95 latency: **180ms (Http) vs 120ms (Fast)** — FastHttpUser is 33% faster at the tail
- p99 latency: **240ms (Http) vs 240ms (Fast)** — converges at the extreme tail
- The HttpUser 1000-user test even threw a **CPU usage warning** ("CPU usage above 90%"), while FastHttpUser did not. This is the key insight: at high concurrency, HttpUser's Python `requests` library consumes more CPU on the local machine, which inflates response times. FastHttpUser's C-optimized parser handles the same load with less overhead.

5. Why no difference at low concurrency? At 10-100 users, both clients are mostly idle between requests (due to `wait_time`). The bottleneck is network latency (~15ms), not local CPU. FastHttpUser's efficiency only matters when the local machine is generating hundreds of requests per second.

6. GET vs POST performance is nearly identical — both have the same median. This is because the in-memory hashmap operations (read or write) are sub-microsecond. The `RWMutex` benefit would become visible with a slower data store (like a database) where read locks allow true parallelism while write locks block.

7. RPS scaled linearly — 5 → 49 → 248 → 497 RPS as users increased 10 → 100 → 500 → 1000. The server hasn't hit its ceiling yet. We'd likely need thousands more users to find the breaking point.