LOCUST

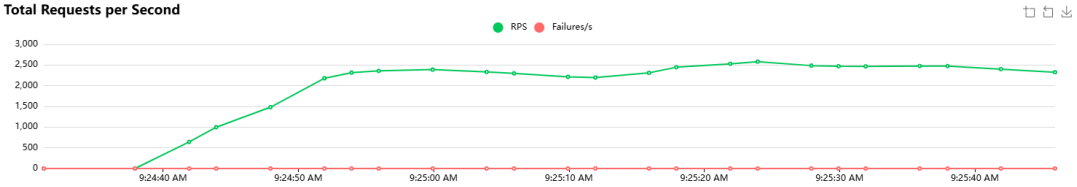
Host	Status	Users	RPS	Failures
http://54.186.18.45:8080	RUNNING	100	2400.7	0%

EDIT LOADING RESET

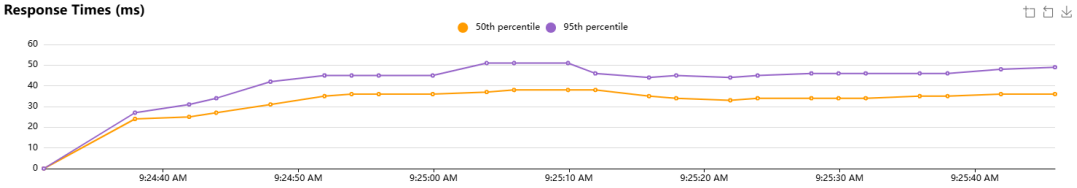
STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS LOCUST CLOUD

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	113507	0	35	46	58	35.75	22	132	107	1800.4	0
POST	/products/:id/details	37627	0	35	46	58	35.87	22	128	0	600.3	0
	Aggregated	151134	0	35	46	58	35.78	22	132	80.36	2400.7	0

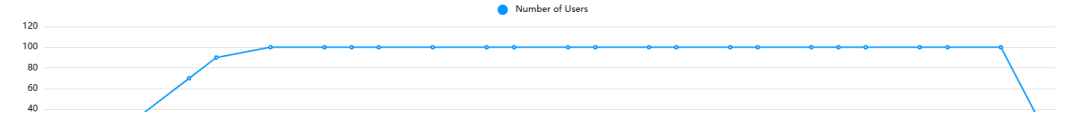
Total Requests per Second



Response Times (ms)



Number of Users



LOCUST

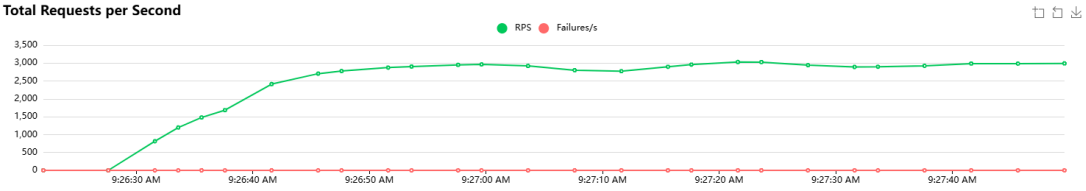
Host	Status	Users	RPS	Failures
http://54.186.18.45:8080	RUNNING	100	2986.4	0%

EDIT LOADING RESET

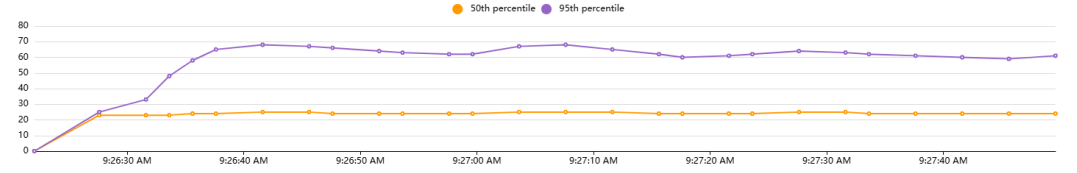
STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS LOCUST CLOUD

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	166774	0	24	63	73	32.49	21	253	107	2236.2	0
POST	/products/:id/details	55649	0	24	63	73	32.7	21	138	0	750.2	0
	Aggregated	222423	0	24	63	73	32.54	21	253	80.23	2986.4	0

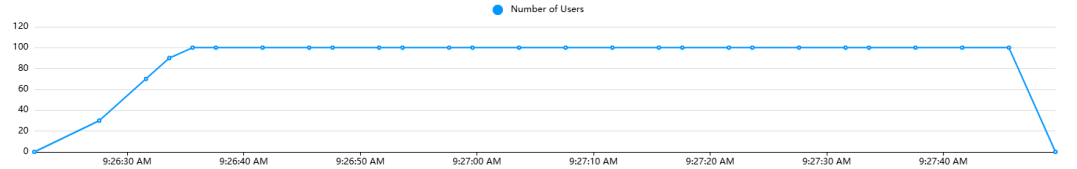
Total Requests per Second



Response Times (ms)



Number of Users



From my experience, when running without any wait time (maximum throughput), FastHttpUser achieves ~20% higher throughput while consuming less CPU compared to HttpUser. This confirms that the C-based asynchronous implementation of FastHttpUser is significantly more efficient for high-concurrency workloads. The standard HttpUser, built on Python's synchronous requests library, incurs more CPU overhead due to blocking I/O and GIL contention and has less RPS.

In the real world, get requests are more used, so using a concurrent read-optimized structure (sync.RWMutex) is ideal.
It allows multiple readers simultaneously while still protecting writes.