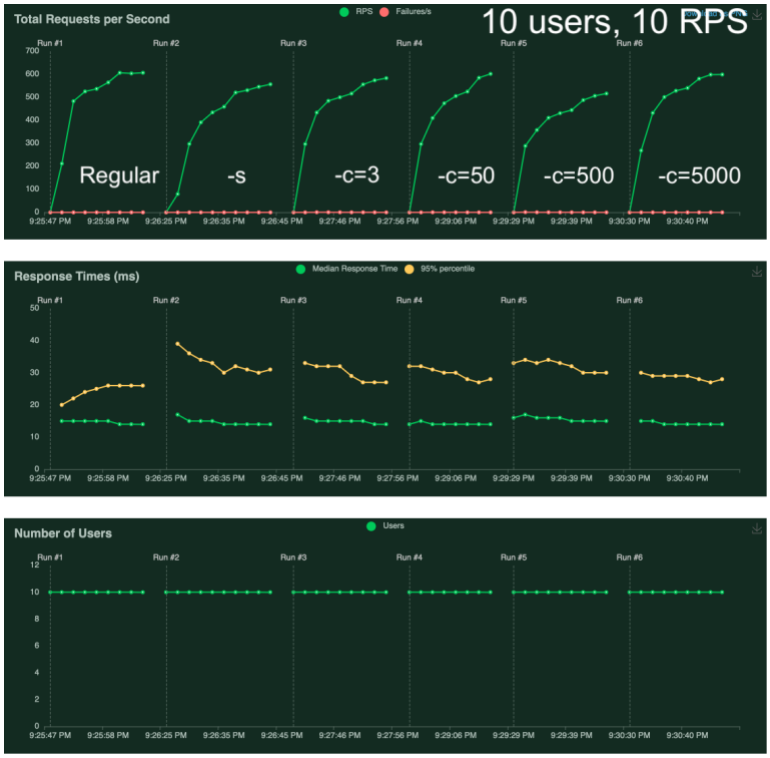
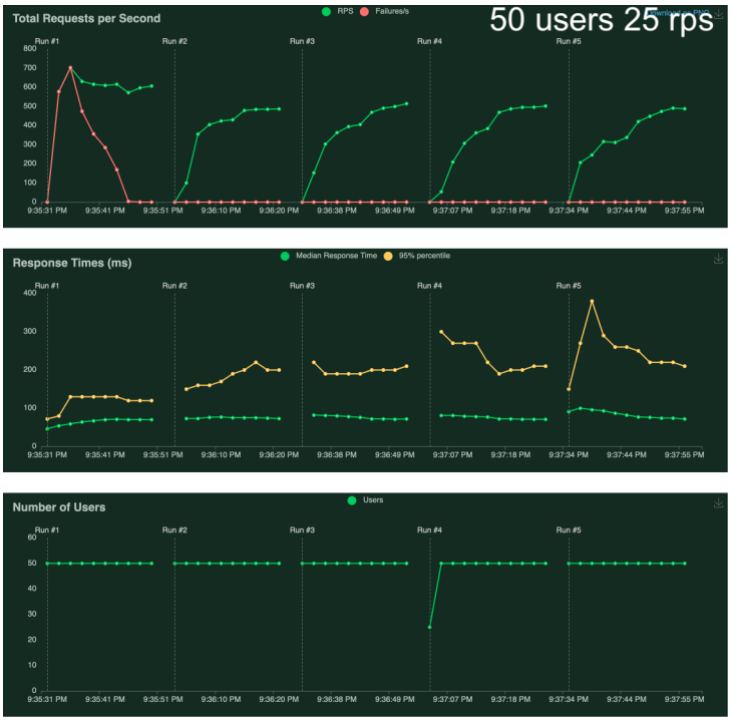
My three benchmark workloads were file100.html, file1000.html, and file10000.html.

```
from locust import HttpUser, task

class WebsiteUser(HttpUser):
@task(2)
def f100(self):
self.client.get("/file100.html")

@task
def f1000(self):
self.client.get("/file1000.html")
```

In my first tests, I swarmed with 10 users and 10 RPS. The results looked about the same for every variation of the webserver in this test with just slight variations that were likely within the margin of error. The response time, however, was slightly slower in all of the non-basic implementations.



In my next tests, I swarmed with 50 users and 25 RPS and got more revealing results. The basic implementation had a lot of failures to begin with and the slowly cleaned itself up. The streaming test functioned with basically no failures and only a slightly increased response time. Caching with a small in like 3 created negligible difference from streaming, however, caching with larger amounts had differences. When n = 50, the response time was slower, but the RPS were about the same. In the n = 500 test, the response time was much slower, but the RPS were more carefully spaced out.



To conclude, the higher the workload that you're putting on your webserver, the more likely it is you're going to want to use a form of caching or streaming. Additionally, you are probably going to want to decide what is more important to you. If you prefer faster response times, then maybe you should go for the streaming option since it is relatively fast and with few failures. If you prefer more requests per second, then you're probably going to want to go for a decently large size n for caching, depending on what's most suitable for your case. In the case of my tests, I think -c=500 was the best option because it was able to slow the RPS and return to a stable level after the initial spike in response times. I'm sure I could've gotten better results had I done more testing.