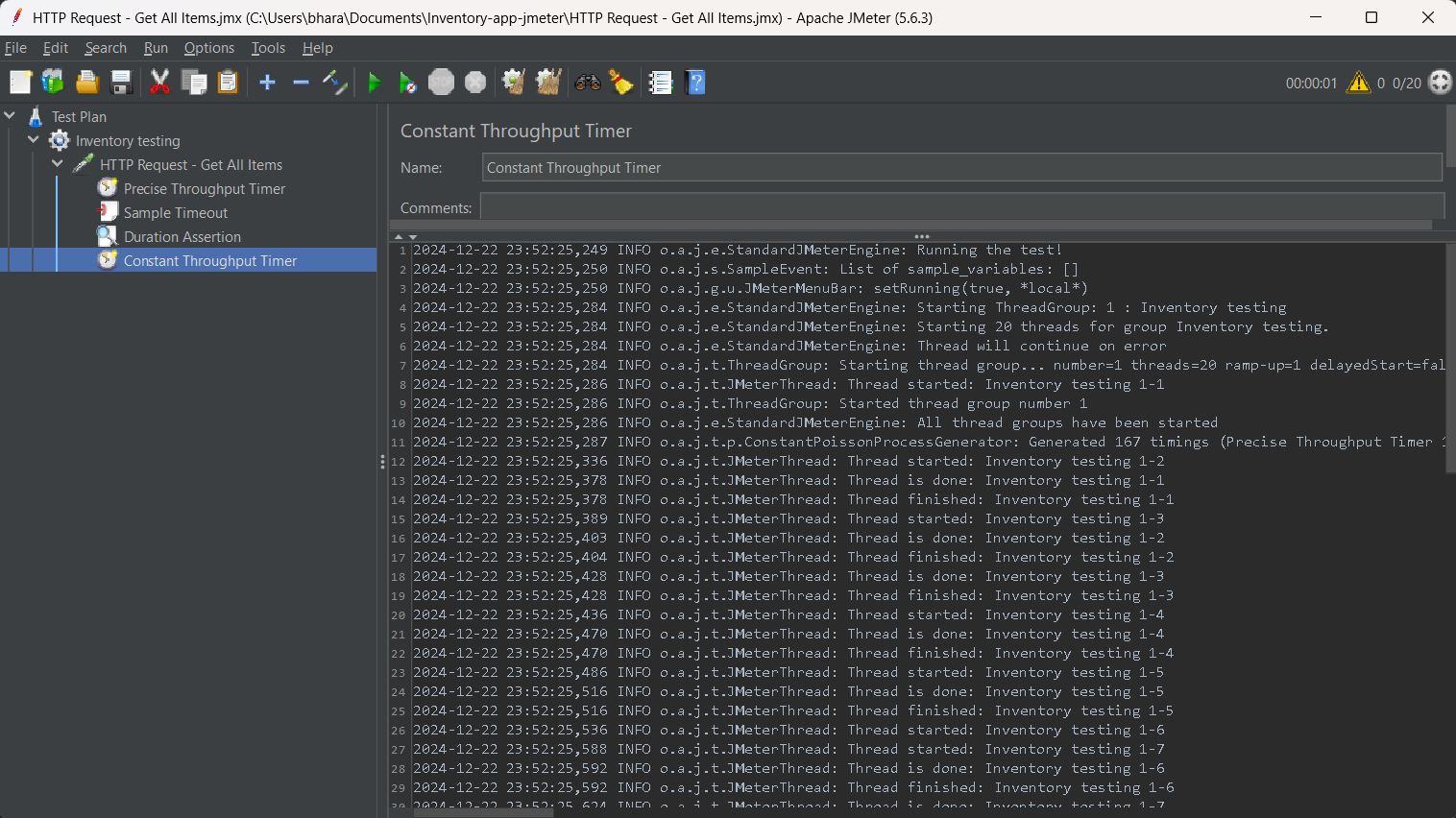
**Inventory Service Performance Report**

Below is the API performance metrics generated using **Jmeter**. We have created an inventory service which supports CRUD operations to add/update/delete details of items present in Inventory.

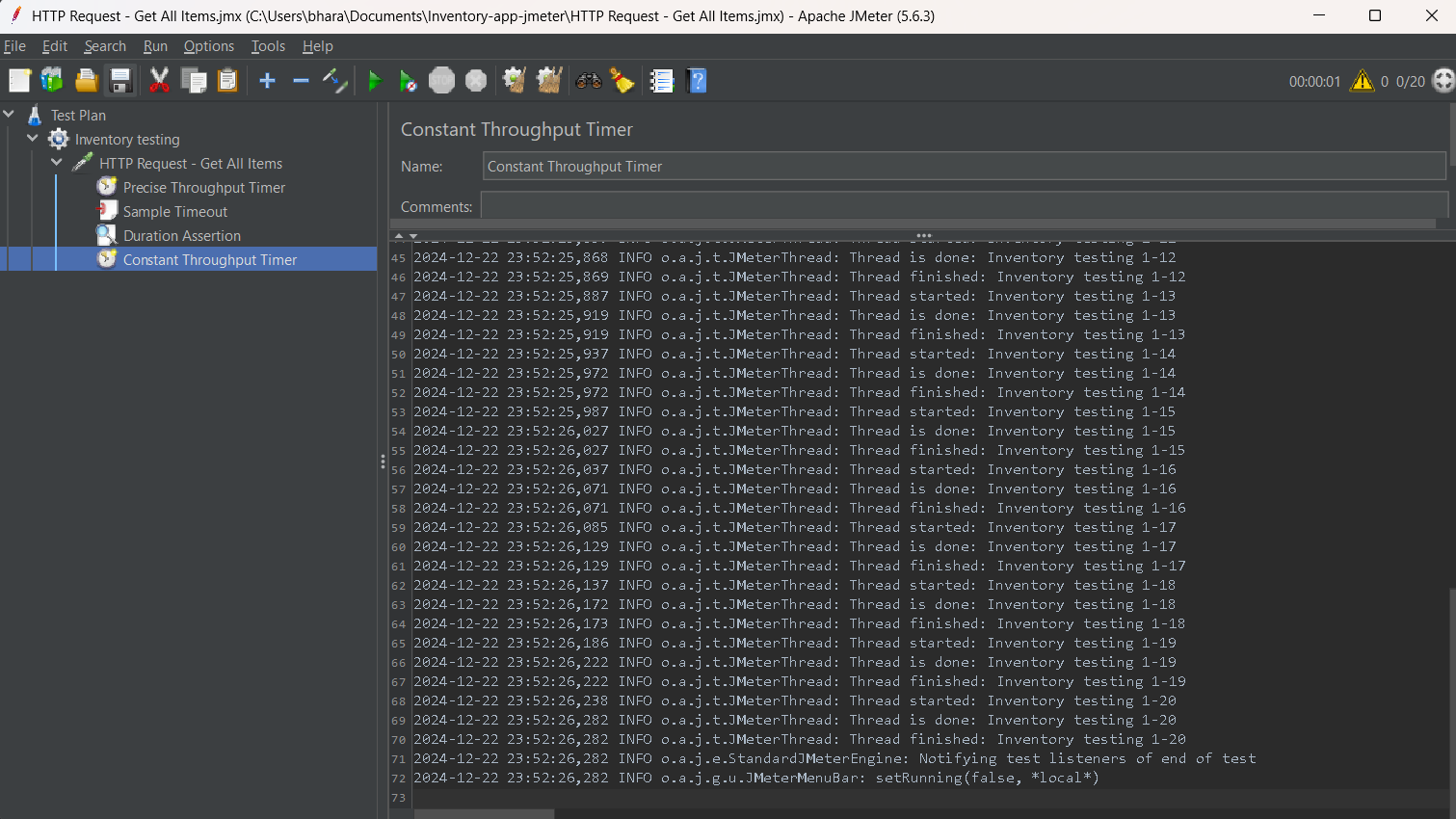
Here we are calling **GetAllItems** endpoint to get all the item details, which mostly uses cached version to return the response.

API Response Statistics using Redis cache **without multi-threading or Async tasks** (**20 concurrent users** configured in Jmeter)

Test Started at 23 : 52 : 25. 249



Test finished at 23:52:26.282 (took around 1s)

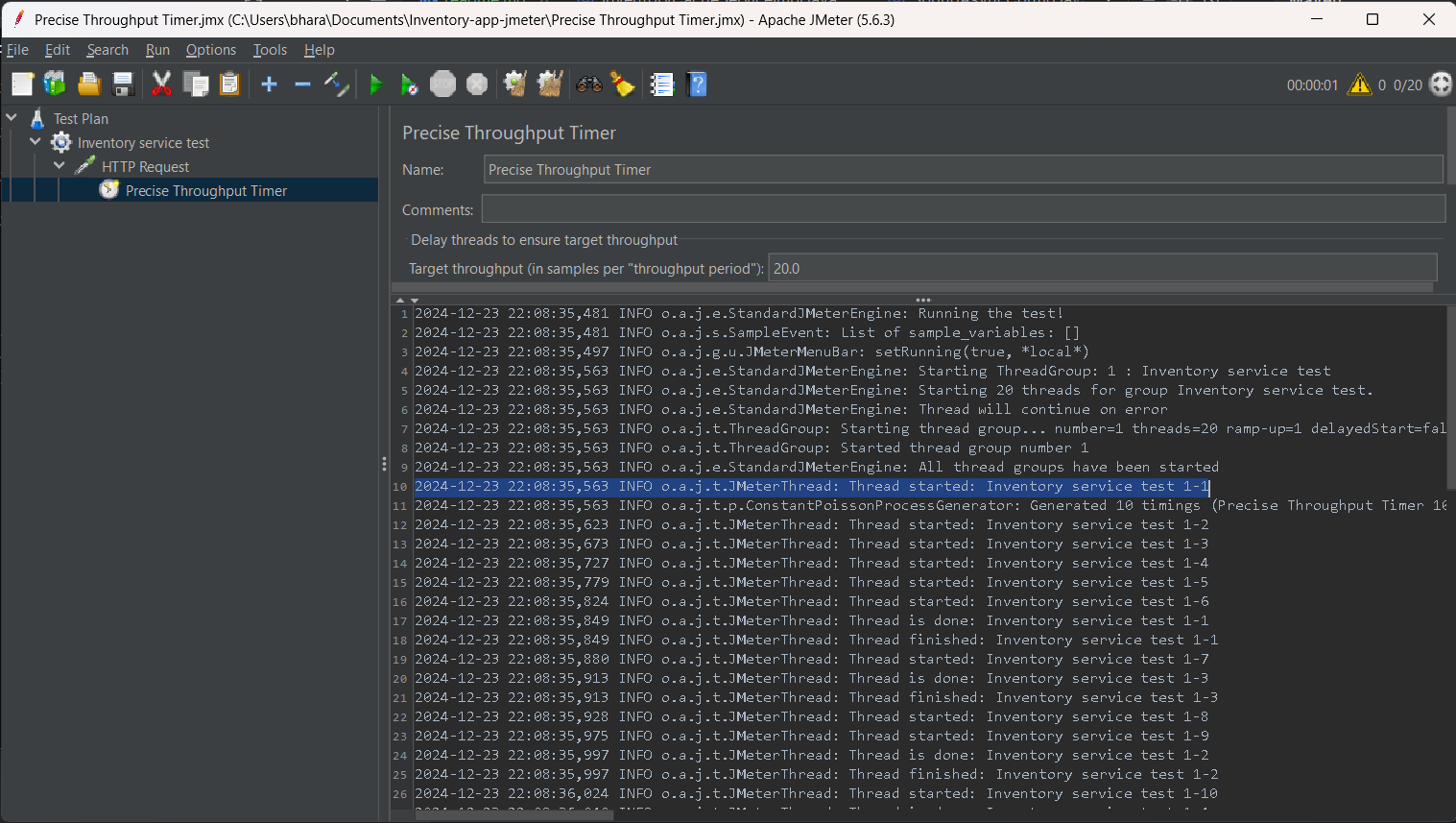


Average response time : **50ms**

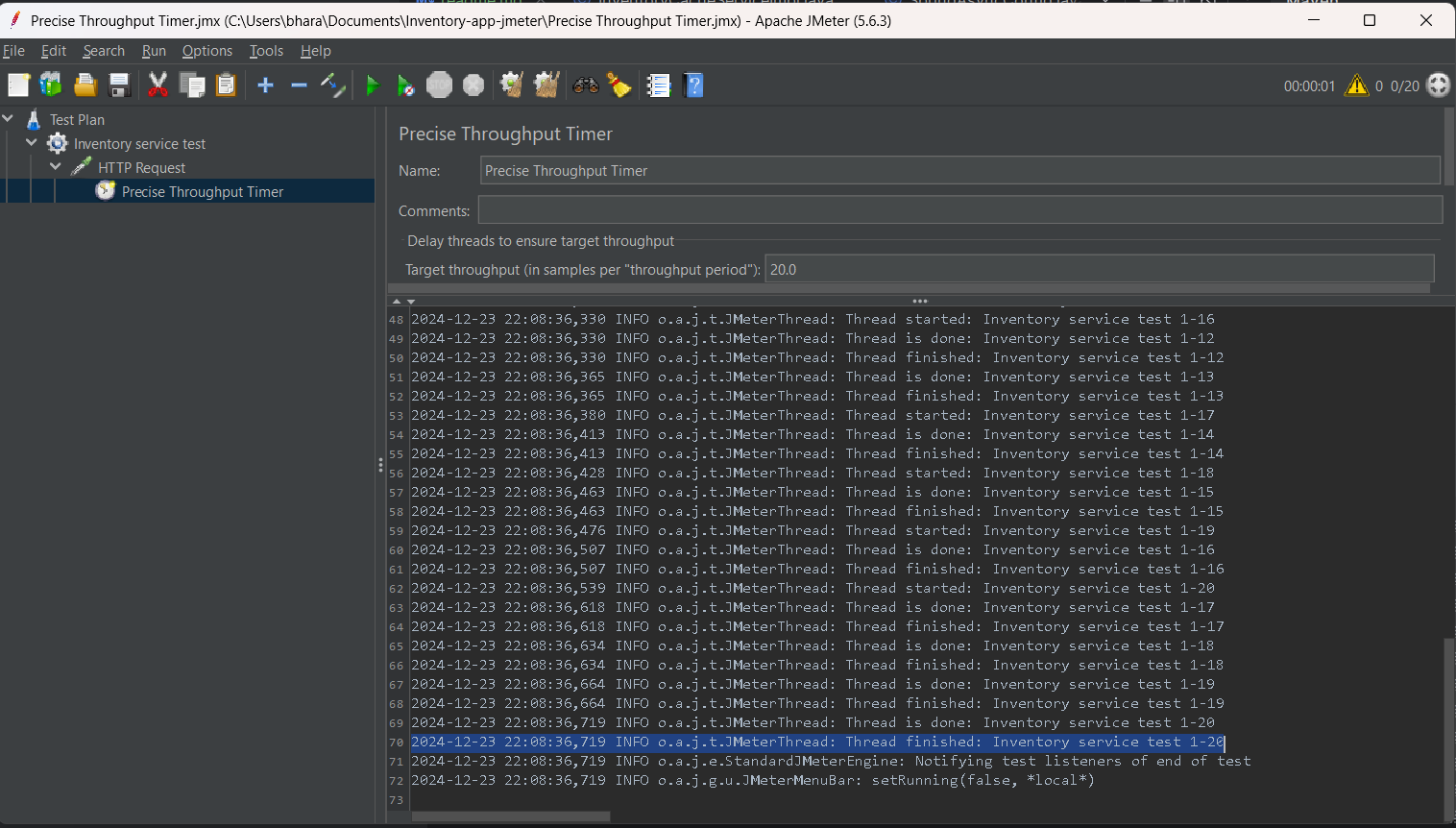
API Response Statistics using Redis cache **with multi-threading or Async tasks** and below thread pool executor configuration :

Core Pool Size = 10   
Max Pool Size = 12   
Queue Capacity = 20

Test Started at 22:08:35.563



Test Finished at 22:08:36.719 (around 1.1s)

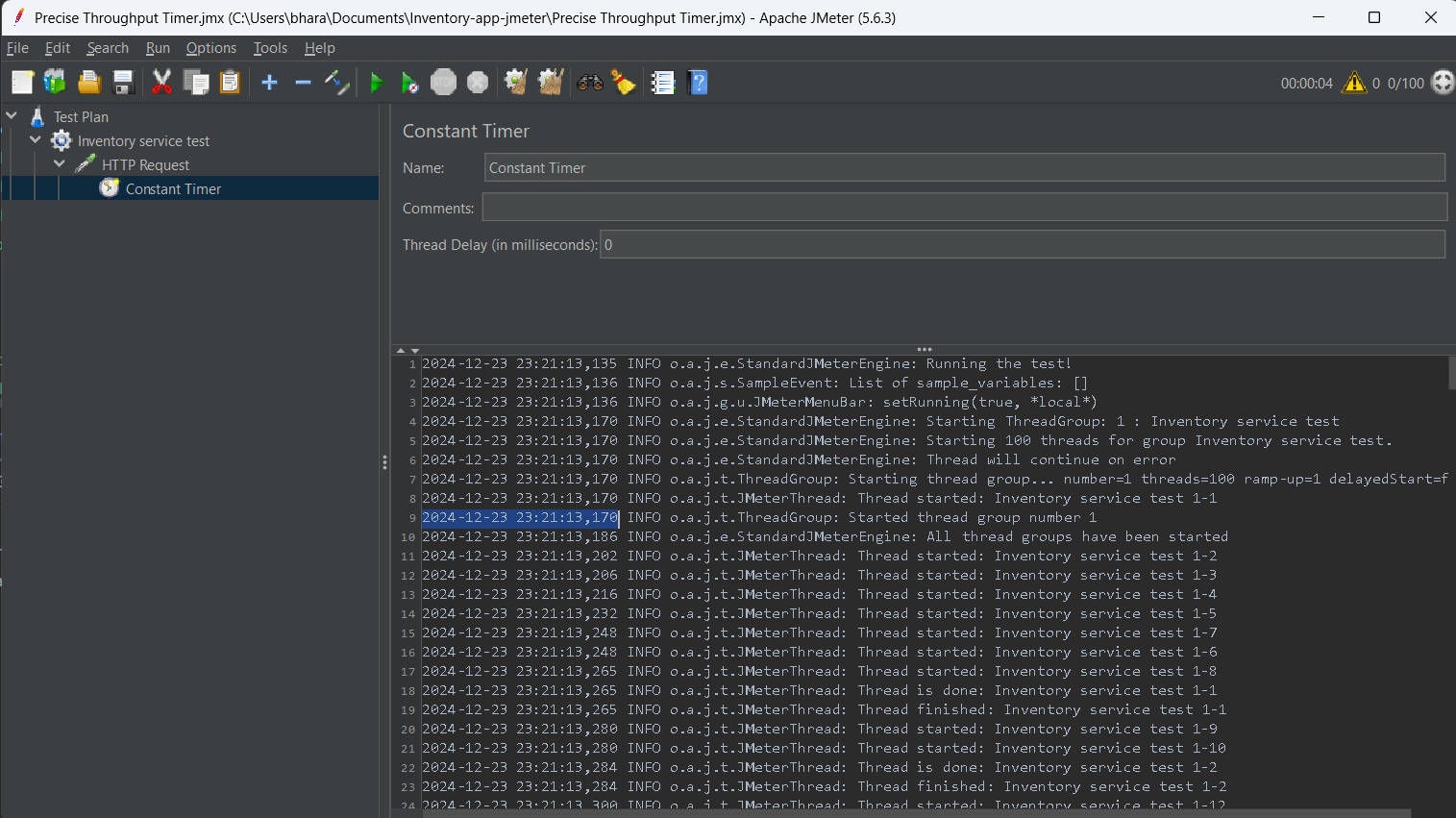


Average response time : **55ms**

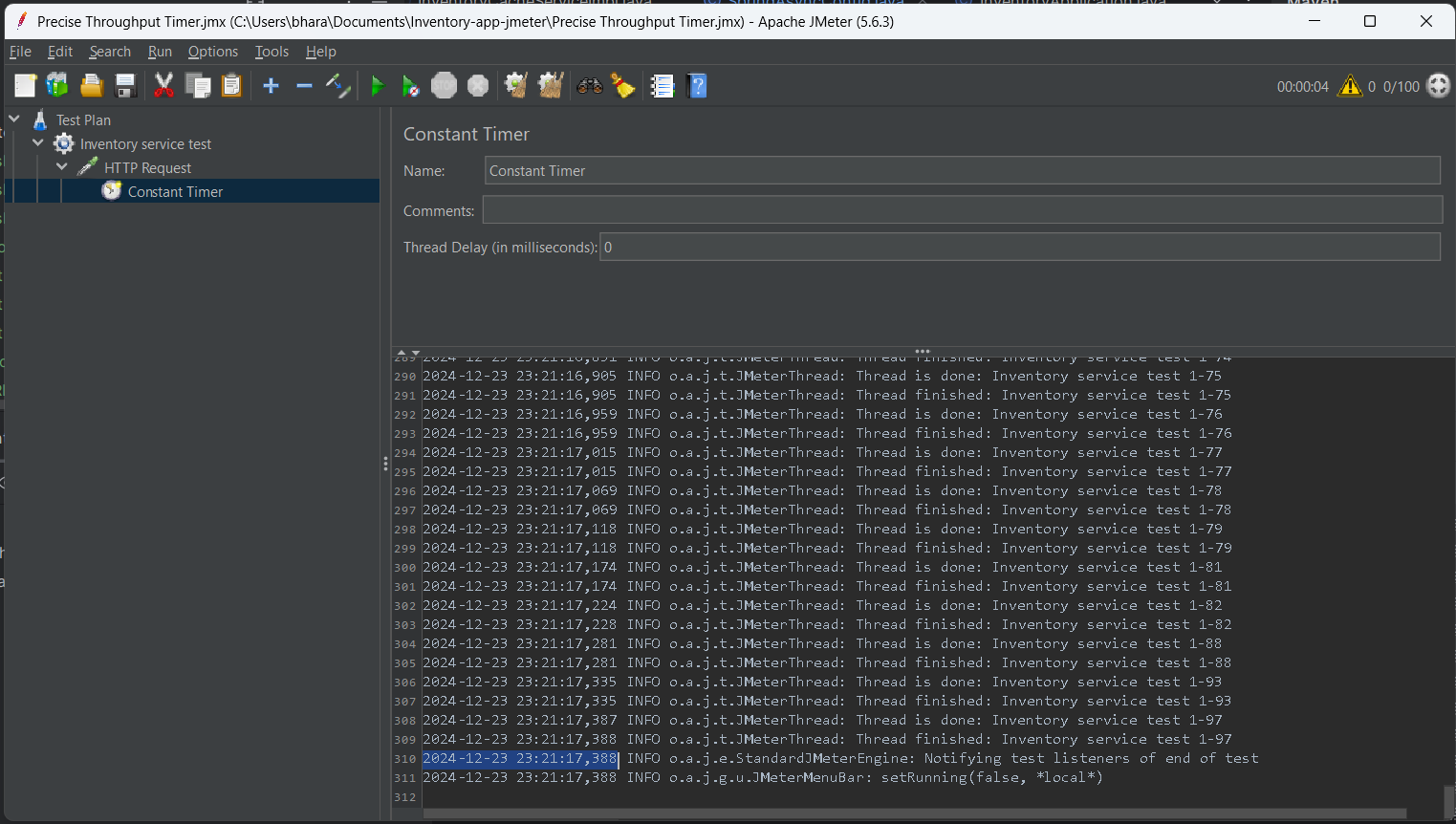
With **100 concurrent requests** total through put time of **4.2s** was achieved

Also changed the queue size to 100, so that requests don’t get rejected in executor

Test started at 23 : 21 : 13.170



Test Finished at 23 : 21 : 17.388



Average response time : **42ms**

**Conclusion**

Performance did change to some extent but not much with Async tasks, as Spring already tries to run every request in new thread if available. There might be room for improvement that I want to explore, may be application utilizing single Redis connection, for the given time I was not able to explore on that use case. But for this scenario and setup I was not able to see significant difference with Async.