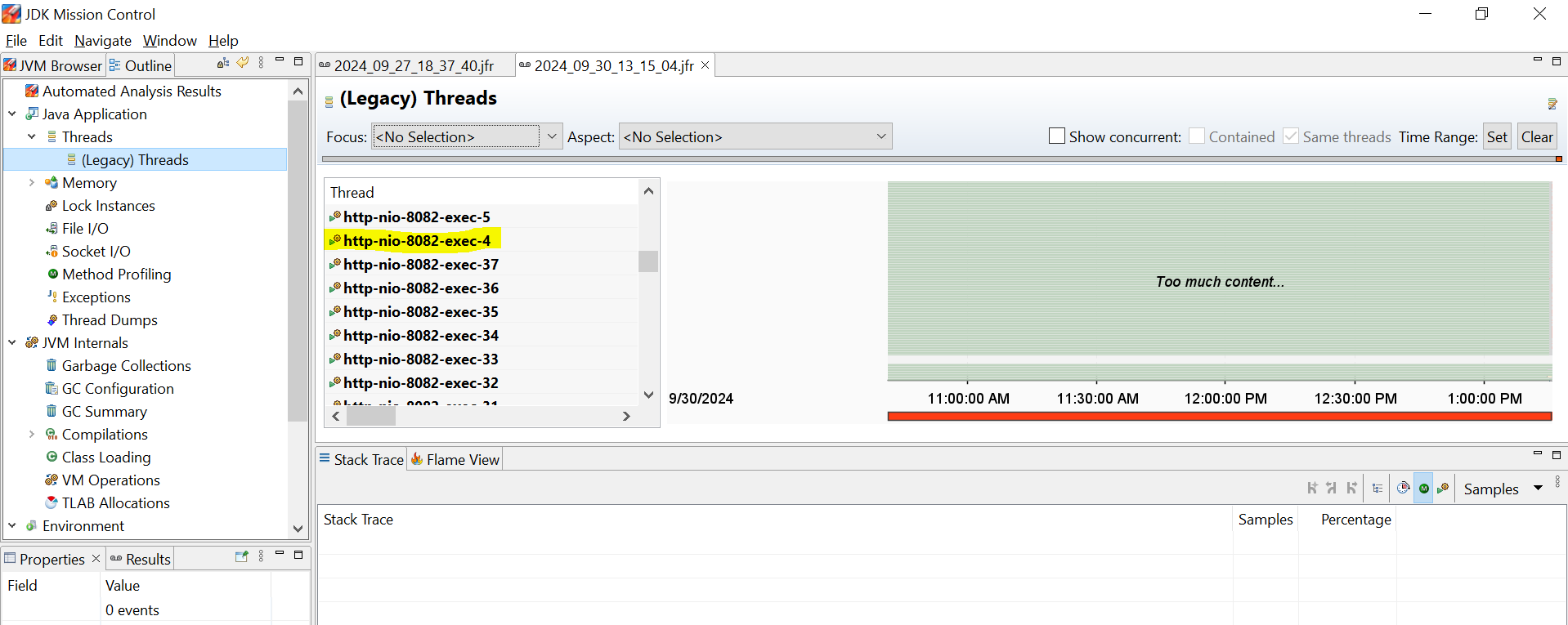
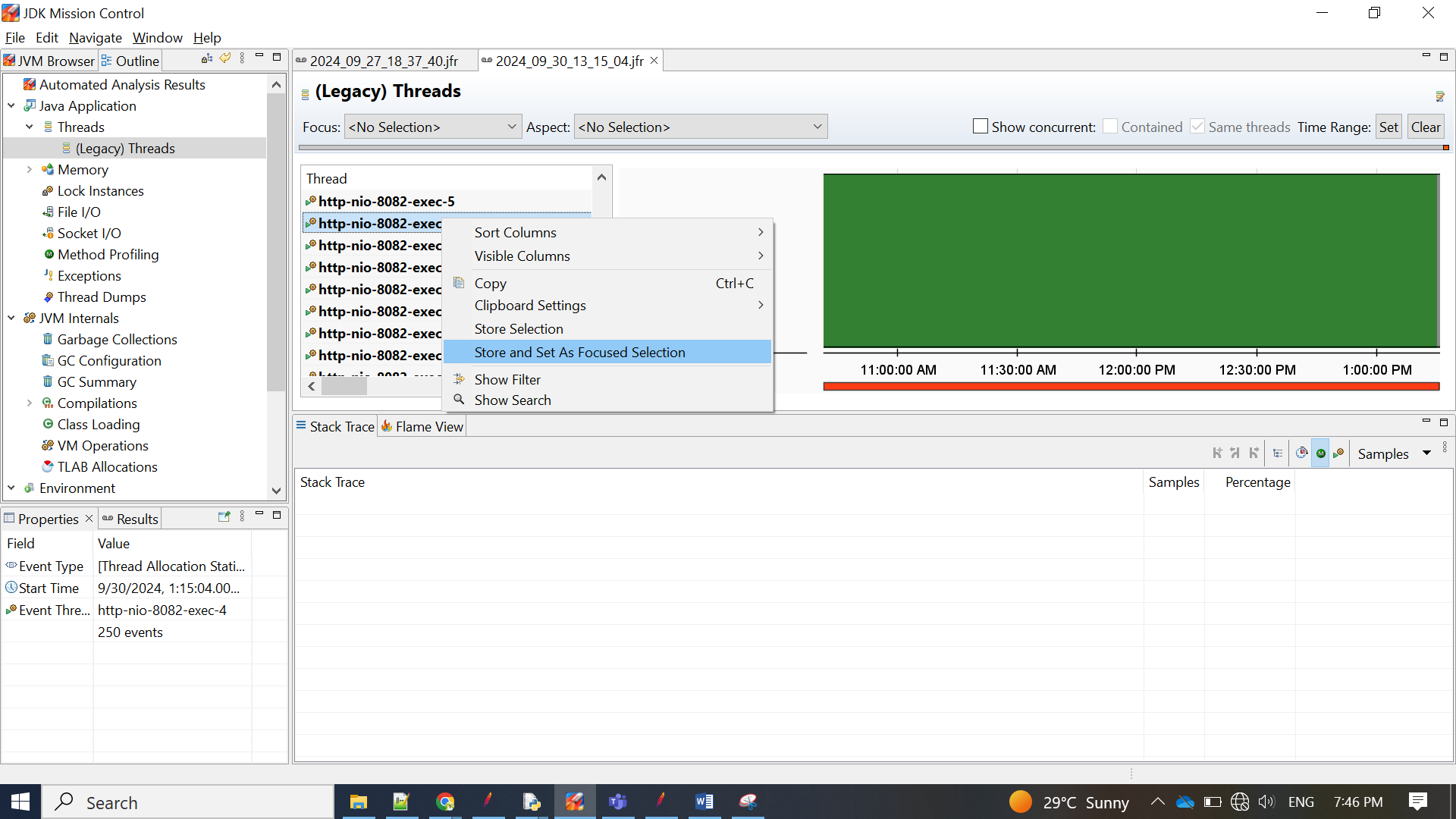
**JFR Analysis for Promise Engine:**

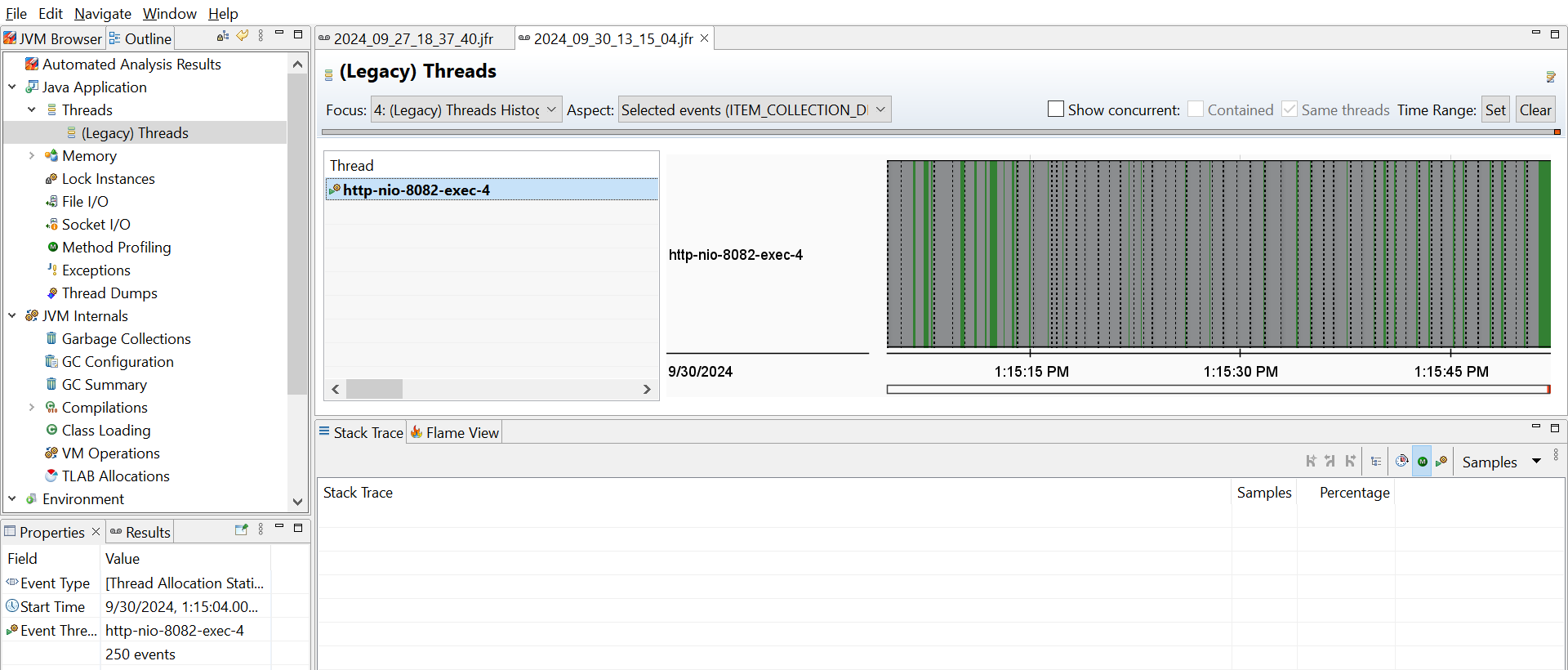
JFR Analysis for Promise Engine “v1/tat/sku” API call transaction High Response Time steps and analysis details:

1. Collect the High response time sample transactions from access log using following command or with any other feasible commands from the server.
   1. grep "/v1/tat/sku" access-log.2024-09-30.log| awk '$6 >= "13:02" && $6 <= "13:18"' | awk '{print $8, $5, $6, $9, $10, $16, $17, $19}' |sort -k1,1g | tail -20
   2. sample access logs are:
      1. 1395 [30-09-2024 13:15:12.460] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-6 [3143a4ed-f322-43f4-b786-369d54c16871]
      2. 1398 [30-09-2024 13:15:06.645] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-13 [17d46091-c33d-44f7-9a19-d909a46469a8]
      3. 1402 [30-09-2024 13:15:06.666] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-17 [3439d7ff-6140-4171-ae2a-019a8b10dc65]
      4. 1404 [30-09-2024 13:15:14.867] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-3 [32839cc4-a32a-4ea0-bd09-2fefabd4f540]
      5. 1404 [30-09-2024 13:15:14.869] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-7 [3b64e209-b38f-4a53-b9a5-f66383fb66cb]
      6. 1414 [30-09-2024 13:15:13.334] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-15 [0b38a572-207b-44b6-9701-0c2142f164e7]
      7. 1422 [30-09-2024 13:15:07.027] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-23 [d42b82c3-938b-4cbf-9adb-6a55f1e77e90]
      8. 1474 [30-09-2024 13:15:12.226] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-16 [79441e42-ea70-4270-85d9-2be1cb770921]
      9. 1476 [30-09-2024 13:15:13.320] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-32 [d0606ef0-0751-476f-8f2e-ee571b8b6ebf]
      10. 1479 [30-09-2024 13:15:13.130] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-14 [08b22b29-498d-4163-8029-afdddbf382b4]
      11. 1481 [30-09-2024 13:15:12.835] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-29 [99a2b84b-4abf-4c88-a6c9-e8f04144c9cb]
      12. 1481 [30-09-2024 13:15:13.522] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-9 [5b1cc409-7f8c-4822-b417-e4edb2b8374d]
      13. 1482 [30-09-2024 13:15:14.729] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-14 [fb58de42-360b-4ad6-afa2-87f94f1ab2cb]
      14. 1497 [30-09-2024 13:15:09.348] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-1 [efc1b2ab-4f39-4c33-a8d0-5de5af24bdec]
      15. 1502 [30-09-2024 13:15:13.356] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-31 [519da592-8e7e-4e85-a9bb-350d3fba9332]
      16. 1508 [30-09-2024 13:15:13.236] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-25 [ed386db5-3c28-4905-956a-cc935099c662]
      17. 1540 [30-09-2024 13:15:13.365] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-33 [e8943173-0a8a-432b-945b-401324d9601f]
      18. 1703 [30-09-2024 13:15:13.319] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-23 [13a8ff30-8581-4d6c-9c07-57d2b2bd3d0c]
      19. 1789 [30-09-2024 13:15:13.240] "POST /promise/v1/tat/sku (Java/17.0.3)" http-nio-8082-exec-4 [9ca4ca3a-230f-408d-b312-04658d531e36]
   3. After collect the high response time samples from access log chose the few High RT calls from the collected transactions and subtract the transaction response from the overall transaction time as shown below.
      1. **1789 [30-09-2024 13:15:13.240]** "POST /promise/v1/tat/sku (Java/17.0.3**)" http-nio-8082-exec-4** [9ca4ca3a-230f-408d-b312-04658d531e36]
      2. **1789** 🡪 Response time of the transaction
      3. **[30-09-2024 13:15:13.240]** 🡪 Overall response time of the transaction (End time of the transaction)
      4. **Start time of the Transaction = End time of the Transaction – Response time of the transaction**
      5. Start time = 30-09-2024 13:15:13.240 – 1789
      6. Start time = 30-09-2024 13:15:11.451
2. Collect the JFRs and application logs for the timestamp from the particular server where we collected the access and start analyzing.
3. Frist start the analysis from the application logs and identify at what part of transaction its spent time checking the transaction starting to end.
   1. 2024-09-30 13:15:11,518 INFO {skuInv-fetch-108} [c.r.s.p.dataService.SkuDataService ] request\_id=9ca4ca3a-230f-408d-b312-04658d531e36 time taken by node having inventory for sku 441144824040 by redis 61
   2. 2024-09-30 13:15:11,523 INFO {skuInv-fetch-76} [c.r.s.p.dataService.SkuDataService ] request\_id=9ca4ca3a-230f-408d-b312-04658d531e36 time taken by node having inventory for sku 441144824041 by redis 66
   3. 2024-09-30 13:15:11,527 INFO {skuInv-fetch-84} [c.r.s.p.dataService.SkuDataService ] request\_id=9ca4ca3a-230f-408d-b312-04658d531e36 time taken by node having inventory for sku 441144824039 by redis 70
   4. 2024-09-30 13:15:11,528 INFO {http-nio-8082-exec-4} [c.r.s.p.service.impl.PromiseServiceImpl ] request\_id=9ca4ca3a-230f-408d-b312-04658d531e36 Total time taken for fetch all inventory of node of sku 74
   5. ====> 409ms
   6. 2024-09-30 13:15:11,937 INFO {http-nio-8082-exec-4} [c.r.s.p.service.impl.PromiseServiceImpl ] request\_id=9ca4ca3a-230f-408d-b312-04658d531e36 Allowed node types for fulfillment: [DC, Store]
   7. 2024-09-30 13:15:11,949 INFO {http-nio-8082-exec-4} [c.r.s.p.service.impl.PromiseServiceImpl ] request\_id=9ca4ca3a-230f-408d-b312-04658d531e36 Allowed node types for fulfillment: [DC, Store]
   8. 2024-09-30 13:15:11,959 INFO {http-nio-8082-exec-4} [c.r.s.p.service.impl.PromiseServiceImpl ] request\_id=9ca4ca3a-230f-408d-b312-04658d531e36 Allowed node types for fulfillment: [DC, Store]
   9. ====> 555ms
   10. 2024-09-30 13:15:12,514 INFO {http-nio-8082-exec-4} [c.r.s.p.service.impl.PromiseServiceImpl ] request\_id=9ca4ca3a-230f-408d-b312-04658d531e36 Fetched Ims Batch map from sku Cache : {441144824039=null, 441144824041=null, 441144824040=null} in Calculate TAT
4. In our case time spent on two different transactions
   1. Total time taken for fetch all inventory of node and
   2. Calculate TAT
5. Now open the JFR in JDK Mission control and collect the thread ID from the access log **“http-nio-8082-exec-4”** and searchfor thethread in JFR thread section as shown below.

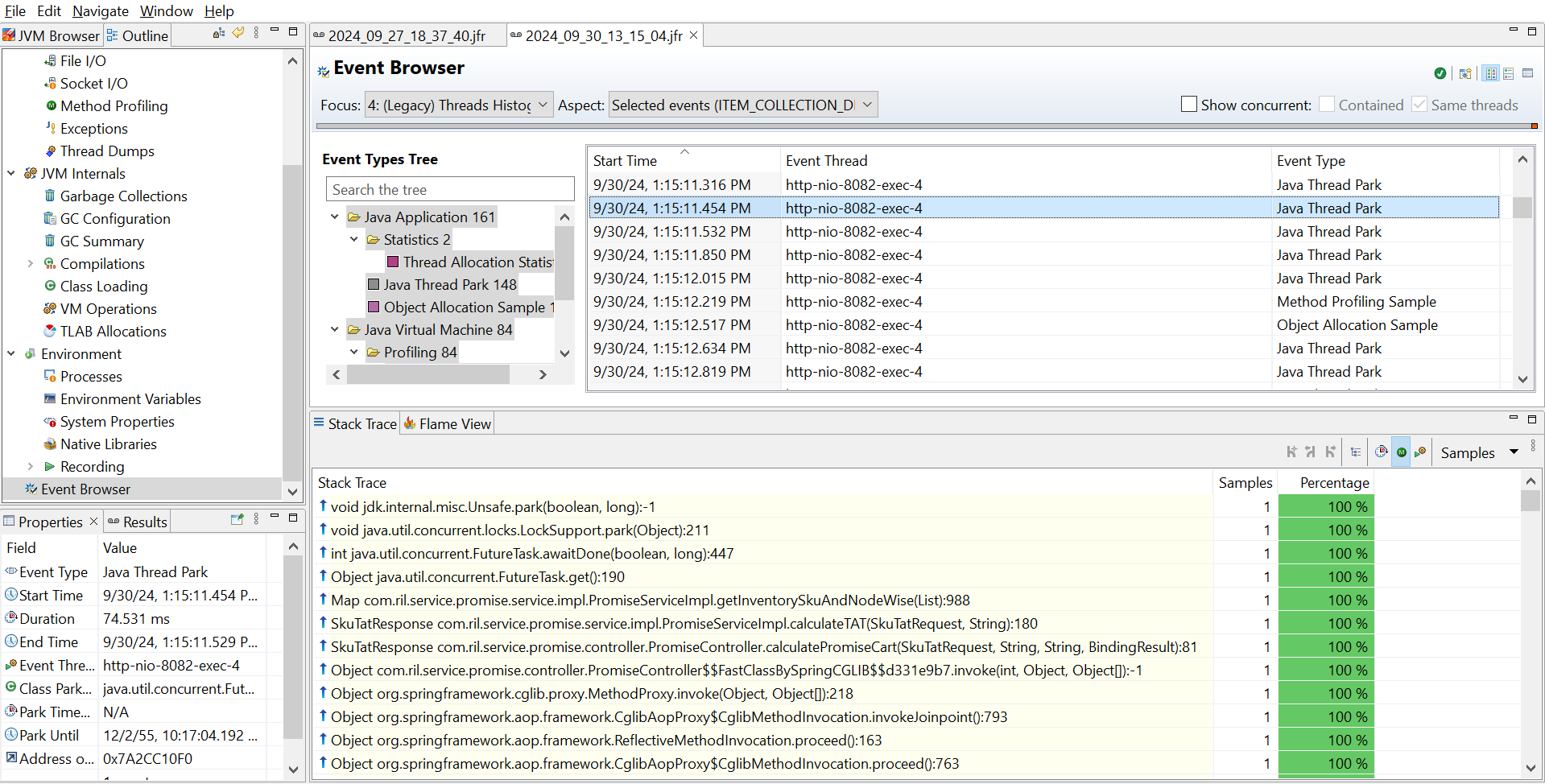


1. Right click on the thread id and select “store and set as focused” option as shown in the below snap.

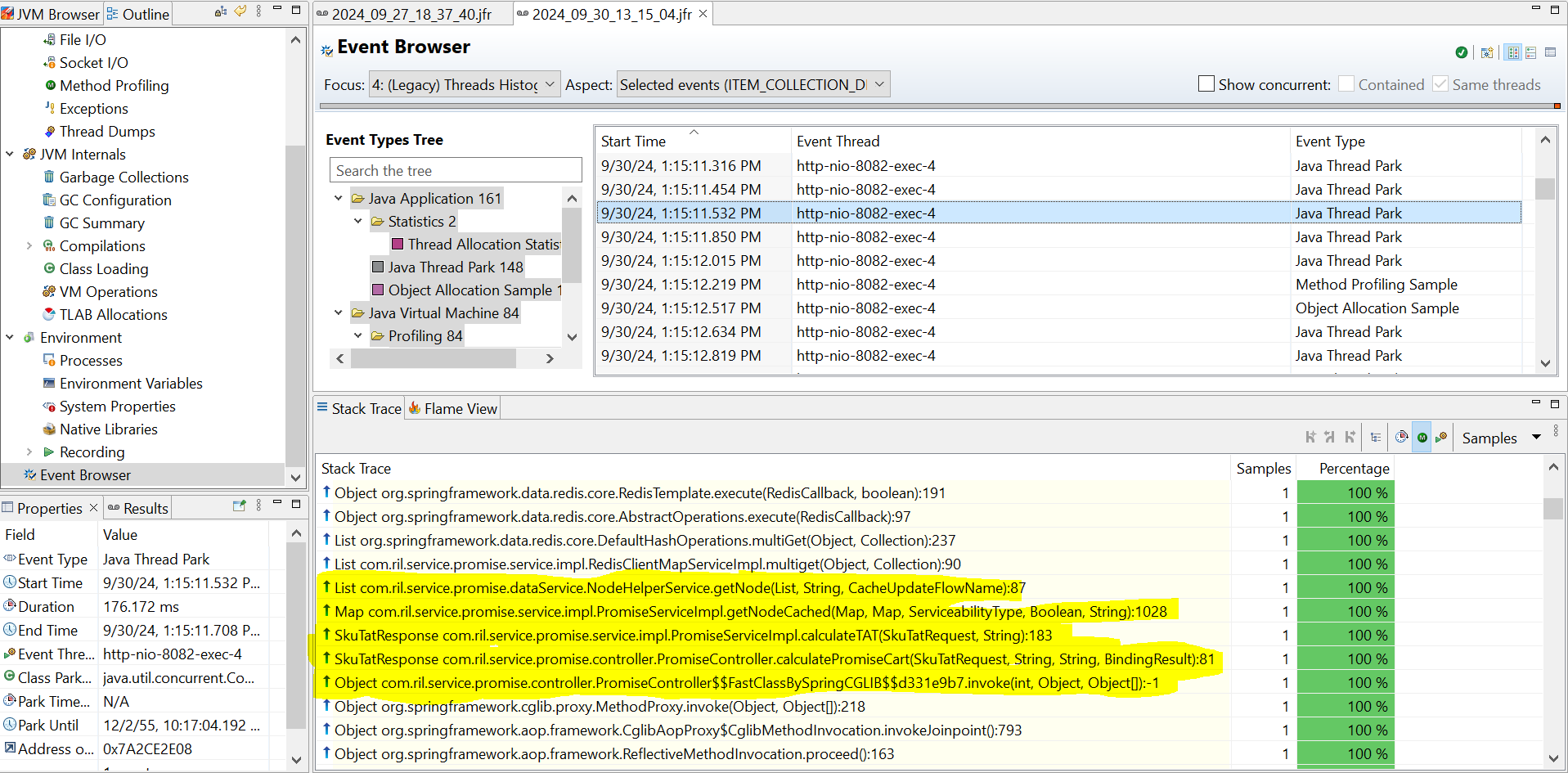


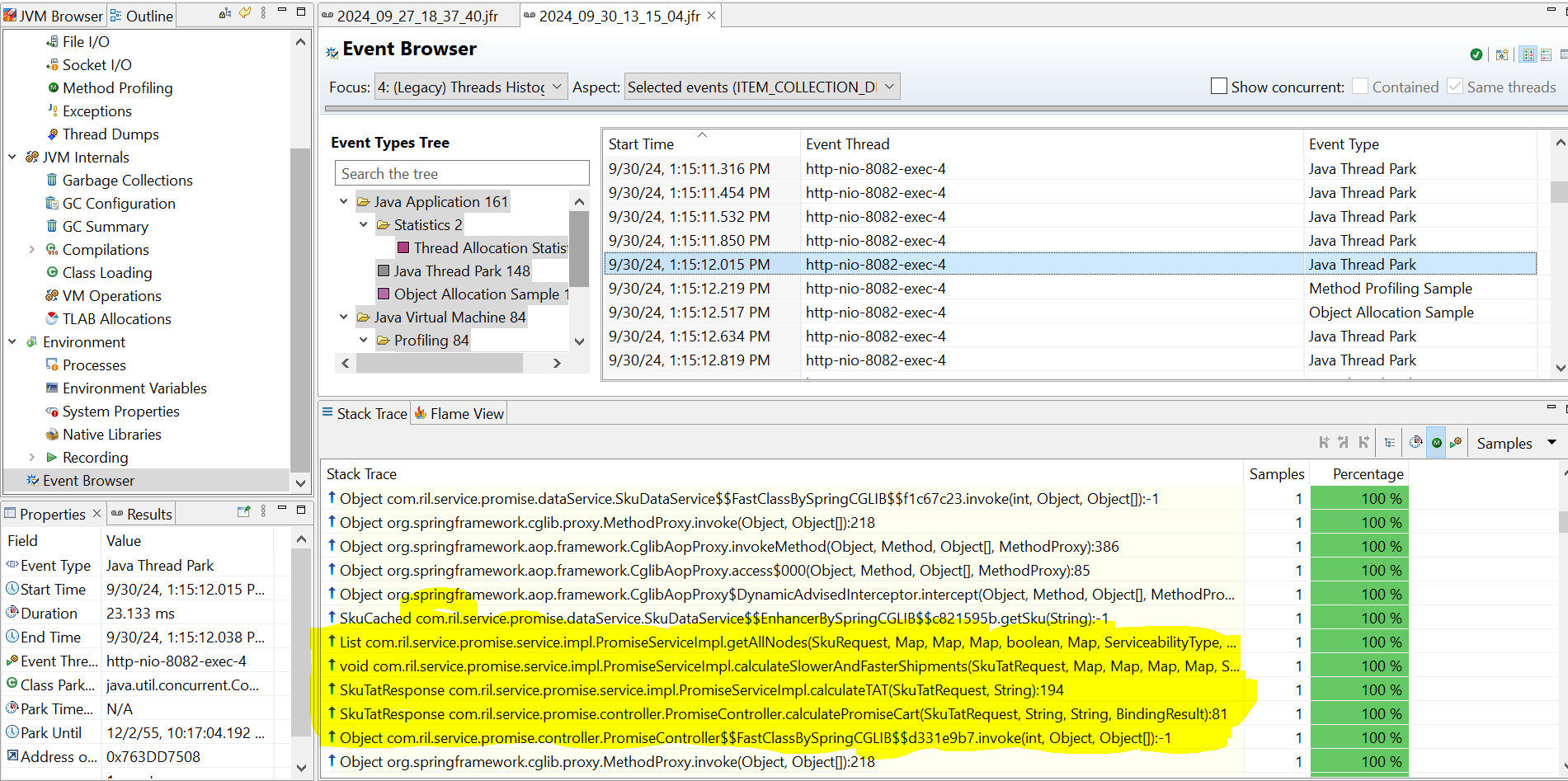


1. Then go to the Event browser and select all the event types tree and sort the even thread based on the start time.



1. Start checking the stack trace from here





1. In this case its spent time on
   1. ===> 409ms **getNodeCached  
      com.ril.service.promise.service.impl.PromiseServiceImpl.getNodeCached**
   2. ===> 555ms **calculateSlowerAndFasterShipments**  
      **com.ril.service.promise.service.impl.PromiseServiceImpl.calculateSlowerAndFasterShipments**
2. Now look into the code base to check further.

