Grab Issue_1528 Report

Problem

Customer filed a ticket "Memory keeps increasing as sending requests - CPU Inference" for TorchServe.

Investigation

- **TorchServe Internal Introduction**

TorchServe allocates a data queue (ie., LinkedBlockingDeque)
for each model. This queue capacity can be configured by setting "`job_queue_size`" in config.properties. Inference request will get error code 503 (ie., SERVICE_UNAVAILABLE) if the inference request is not able to be added into the queue due to the queue full.

- **Test Environment Setting**
    - Profiling Tool: visualVM
    - TorchServe Version: v0.5.3
    - Host: Mac
    - Model: Resnet-18
    - Input: kitten.jpg
    - TorchServe Config

```
inference_address=http://0.0.0.0:8080
management_address=http://0.0.0.0:8081
metrics_address=http://0.0.0.0:8082
#number_of_netty_threads=32
job_queue_size=1000
vmargs=-Xmx128m -XX:-UseLargePages -XX:+UseG1GC
-XX:+ExitOnOutOfMemoryError
prefer_direct_buffer=True
default_response_timeout=300
```

```
unregister_model_timeout=300
install_py_dep_per_model=true
default_workers_per_model=4
```

- **Inference Requests**

Here are two rounds of prediction test results. Each round sends 2000 messages with concurrency 4.

```
ab -q -c 4 -n 2000 -k -p examples/image_classifier/kitten.jpg -T application/jpg
http://127.0.0.1:8080/predictions/resnet-18
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient).....done


Server Software:
Server Hostname:        127.0.0.1
Server Port:            8080

Document Path:          /predictions/resnet-18
Document Length:        174 bytes

Concurrency Level:      4
Time taken for tests:   50.643 seconds
Complete requests:      2000
Failed requests:        0
Keep-Alive requests:    2000
Total transferred:      814000 bytes
Total body sent:        222304000
HTML transferred:       348000 bytes
Requests per second:    39.49 [#/sec] (mean)
Time per request:       101.285 [ms] (mean)
Time per request:       25.321 [ms] (mean, across all concurrent requests)
Transfer rate:          15.70 [Kbytes/sec] received
                        4286.78 kb/s sent
                        4302.47 kb/s total

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:    80  101   5.9    100     157
Waiting:       79  101   5.9    100     156
Total:         80  101   5.9    100     157

Percentage of the requests served within a certain time (ms)
  50%    100
  66%    101
```

```
 75%    103
 80%    104
 90%    108
 95%    111
 98%    116
 99%    119
100%    157 (longest request)


ab -q -c 4 -n 2000 -k -p examples/image_classifier/kitten.jpg -T application/jpg
http://127.0.0.1:8080/predictions/resnet-18
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient).....done


Server Software:
Server Hostname:        127.0.0.1
Server Port:            8080

Document Path:          /predictions/resnet-18
Document Length:        174 bytes

Concurrency Level:      4
Time taken for tests:   49.933 seconds
Complete requests:      2000
Failed requests:        0
Keep-Alive requests:    2000
Total transferred:      814000 bytes
Total body sent:        222304000
HTML transferred:       348000 bytes
Requests per second:    40.05 [#/sec] (mean)
Time per request:       99.865 [ms] (mean)
Time per request:       24.966 [ms] (mean, across all concurrent requests)
Transfer rate:          15.92 [Kbytes/sec] received
                        4347.73 kb/s sent
                        4363.65 kb/s total

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:    84  100   8.0     99     221
Waiting:       84  100   8.0     99     221
Total:         84  100   8.0     99     221

Percentage of the requests served within a certain time (ms)
 50%     99
 66%    100
 75%    102
```

80% 104
90% 111
95% 113
98% 114
99% 115
100% 221 (longest request)

- **Profile Overview**

Figure 1 is the profiling overview. The heap footprint shows that memory gets freed after each test round.

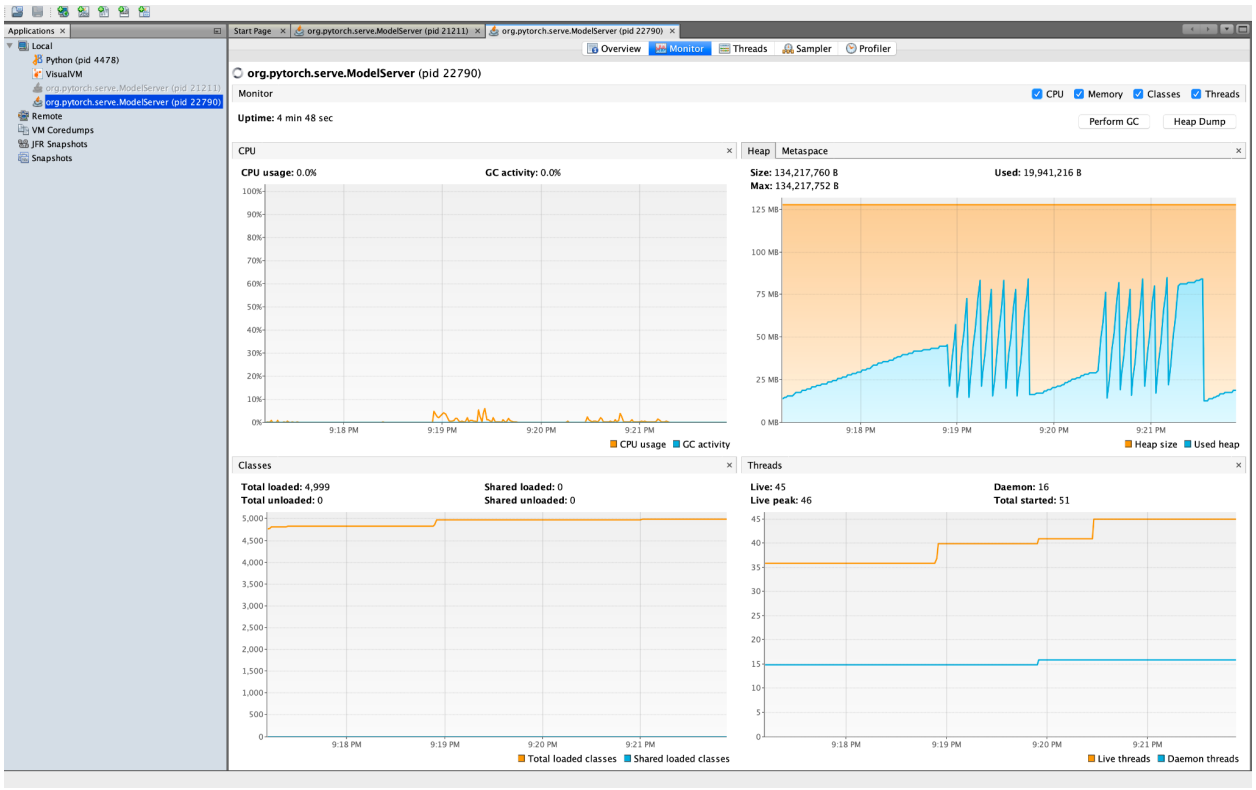

Figure 1: Overview

- **Memory Usage Breakdown**

Figure 2-4 are the memory usage breakdown in each different testing stage. It shows that messages entered into KQueueEventLoopGroup even though there are no incoming requests. KQueueEventLoopGroup is a netty channel for communication b/w frontend and backend, or b/w frontend and client. The allocated memory was also reclaimed (see the waves in the heap graph).

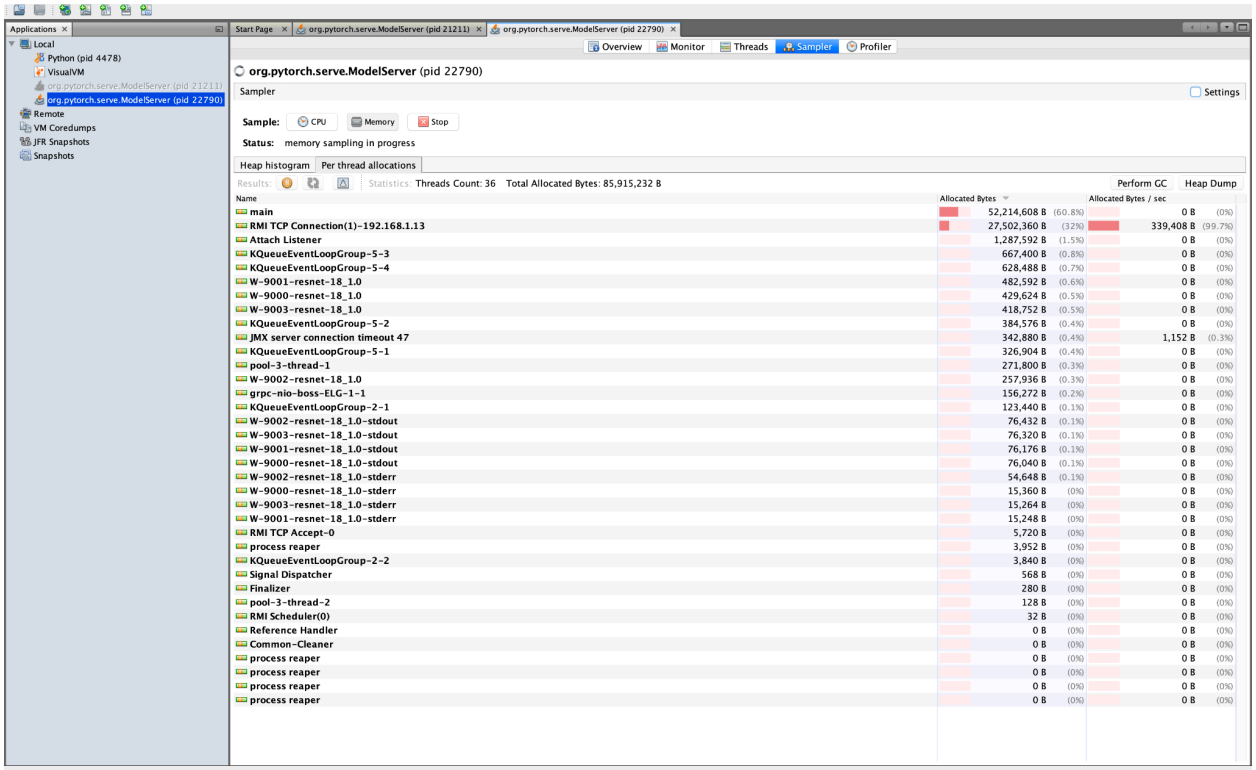

Figure 2: Memory Allocation After Resnet-18 loaded

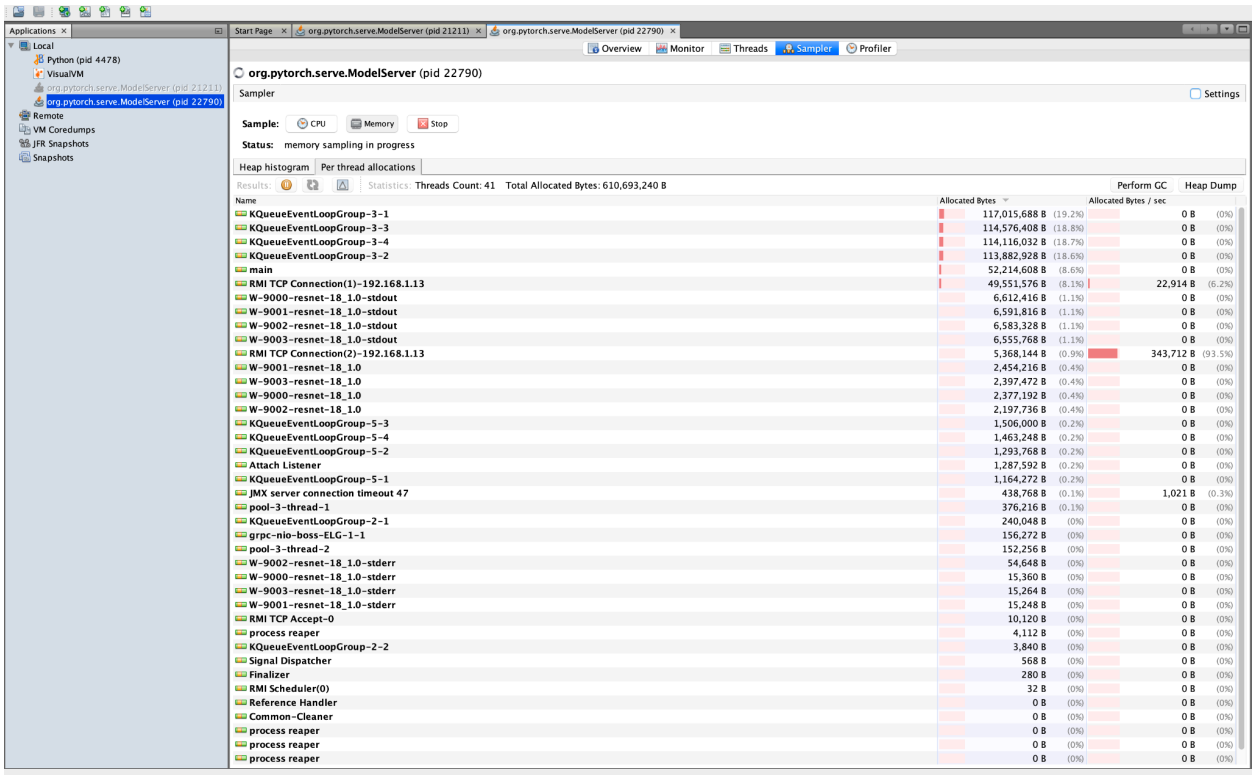

Figure 3: Memory Allocation After the 1st 2000 inference requests

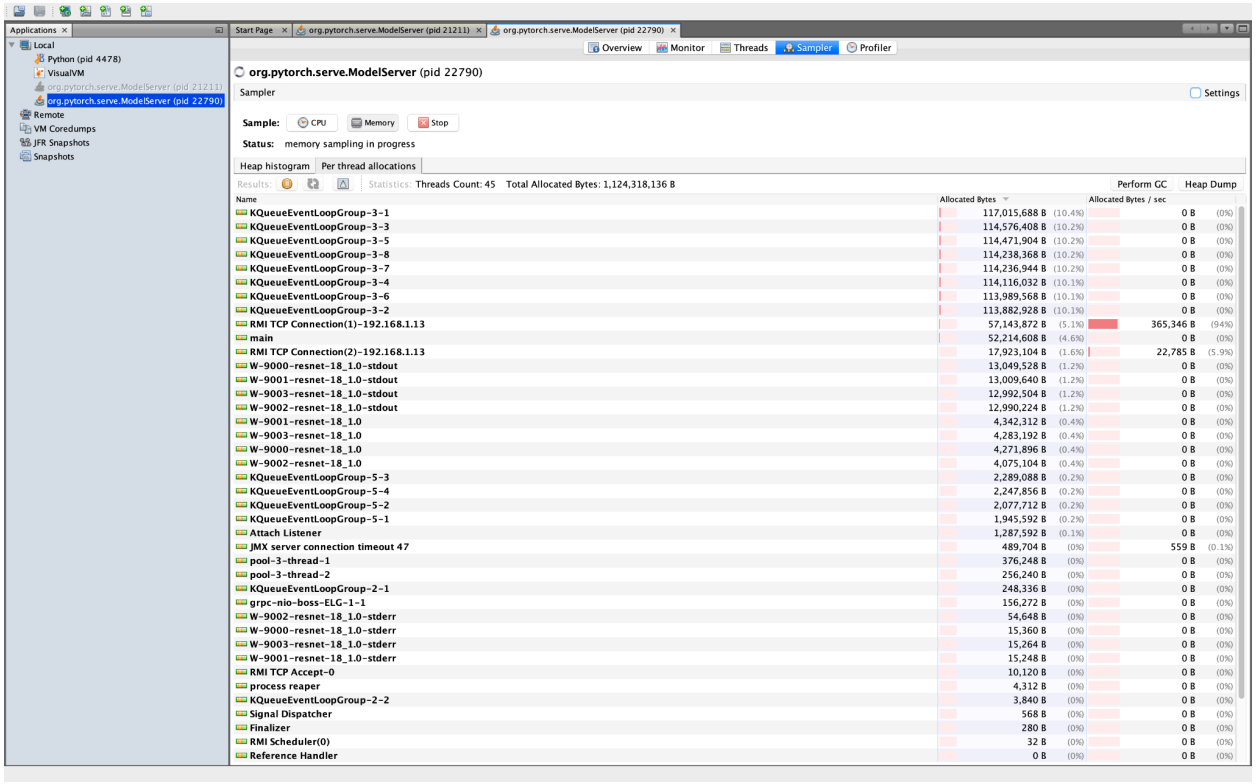

Figure 4: Memory Allocation After the 2nd 2000 inference requests

Summary

- There is no memory leak in TorchServe.
- Provision key factors for TorchServe onboarding production (see configuration)
    - Inference throughput
    - `vmargs (eg.jvm size)`
    - batchSize / maxBatchDelay
    - default_workers_per_model or minWorkers/maxWorkers
    - job_queue_size
    - netty_client_threads
    - number_of_netty_threads
    - async_logging
- Tool

TorchServe provides a benchmark automation tool to help users analyze model latency and system usage.