adaptTo()

EUROPE'S LEADING AEM DEVELOPER CONFERENCE

28th – 30th SEPTEMBER 2020

Sling & Loom

Timothée Maret, Adobe

About the speaker

Timothée Maret

https://twitter.com/timaret

tmaret@apache.org

Sr. Software Developer, Tech Lead @ Adobe R&D

Apache Sling PMC member

adaptTo()

OpenJDK Project Loom

adaptTo()

Loom in a nutshell

- Add virtual threads support for the JVM
  - Supports the java.lang.Thread API
  - Low runtime cost (RAM, scheduling)
  - Java scheduler pluggable to match specific use cases
  - Many virtual threads mapped to few kernel threads

- Available via Early Access builds https://jdk.java.net/loom

Source https://cr.openjdk.java.net/~rpressler/loom/loom/sol1_part1.html

Loom potential

- For services with high wait/compute time ratio
- Do more with less
  - Optimize computational resources usage
  - Increase server throughput
  - Reduce operating costs
- Keep it simple
  - Synchronous programming
  - One virtual thread per task
  - Unbounded or bounded pool

Sling & Loom

Sling is a good use case for Loom

- Provides and relies on blocking APIs
  Resource API, JCR API, etc.

- One thread per request and pool model
  Servlet async processing possible but tricky and limited to few use cases that don’t rely on blocking APIs

- High wait/compute workload ratio
  Request processing requires many repository access

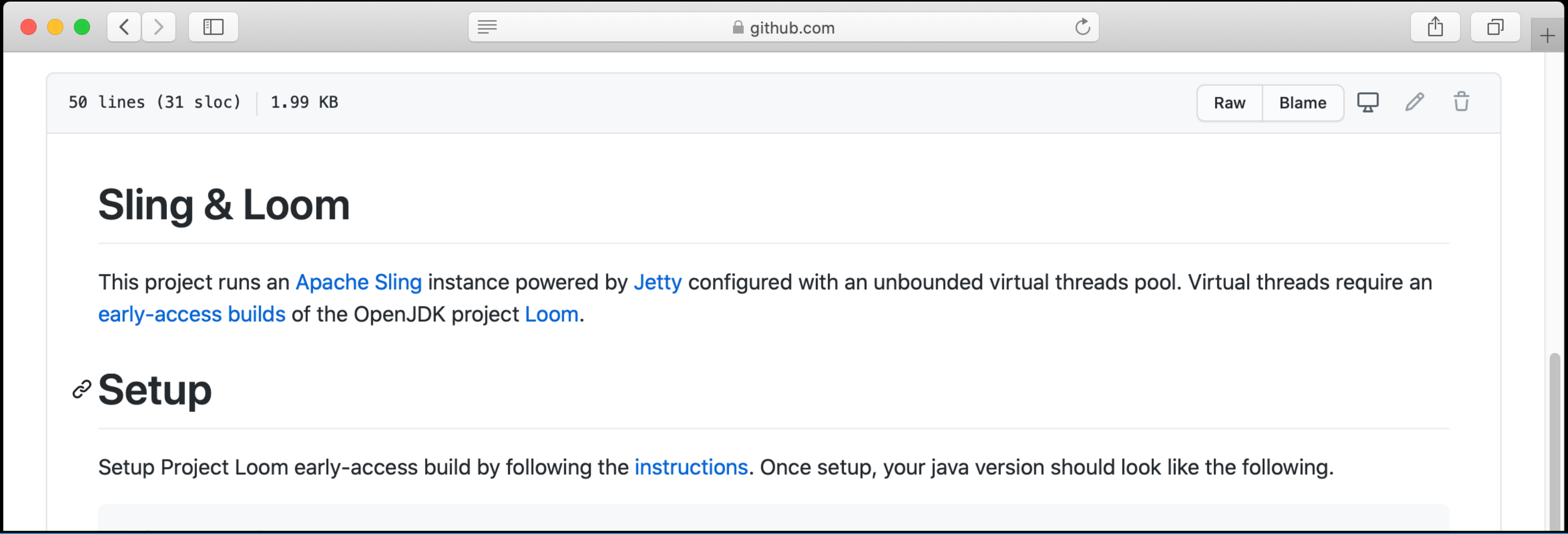
Sling can easily leverage Loom

- Keep same technology stack & application code

- Minor changes required to the Sling framework
    - Plug an unbounded virtual thread pool in Jetty via Apache Felix https://github.com/apache/felix-dev/tree/master/http/jetty
    - Support virtual threads in Sling common threads https://github.com/apache/sling-org-apache-sling-commons-threads
    - Avoid IO in synchronized blocks

Sling and Loom already play together

https://github.com/tmaret/sling-virtual-thread



Sling & Loom

This project runs an Apache Sling instance powered by Jetty configured with an unbounded virtual threads pool. Virtual threads require an early-access builds of the OpenJDK project Loom.

Setup

Setup Project Loom early-access build by following the instructions. Once setup, your java version should look like the following.

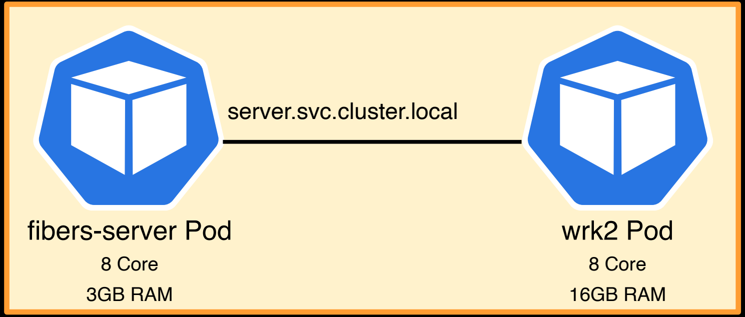
Benchmark

fibers-server

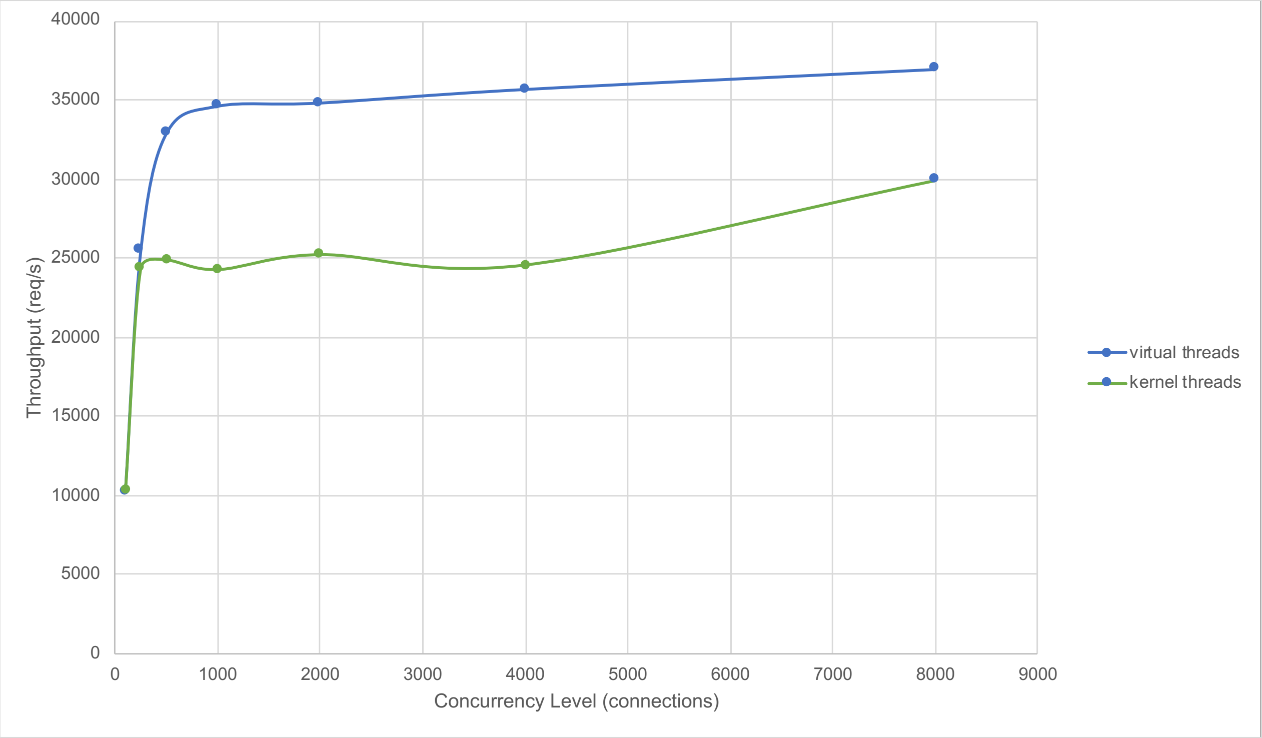
- Jetty based test server
  https://github.com/tmaret/fibers-server
- Configurable workload (wait / compute time)
- Configurable threading support
  - Queued pool of kernel or virtual threads
  - Unbounded pool of virtual threads

Test setup

- 99.5% wait time to compute time ratio
- Two deployments
  - Queue pool with 1000 kernel threads
  - Unbounded pool of virtual threads
- Varies
  - Avg latency & concurrency

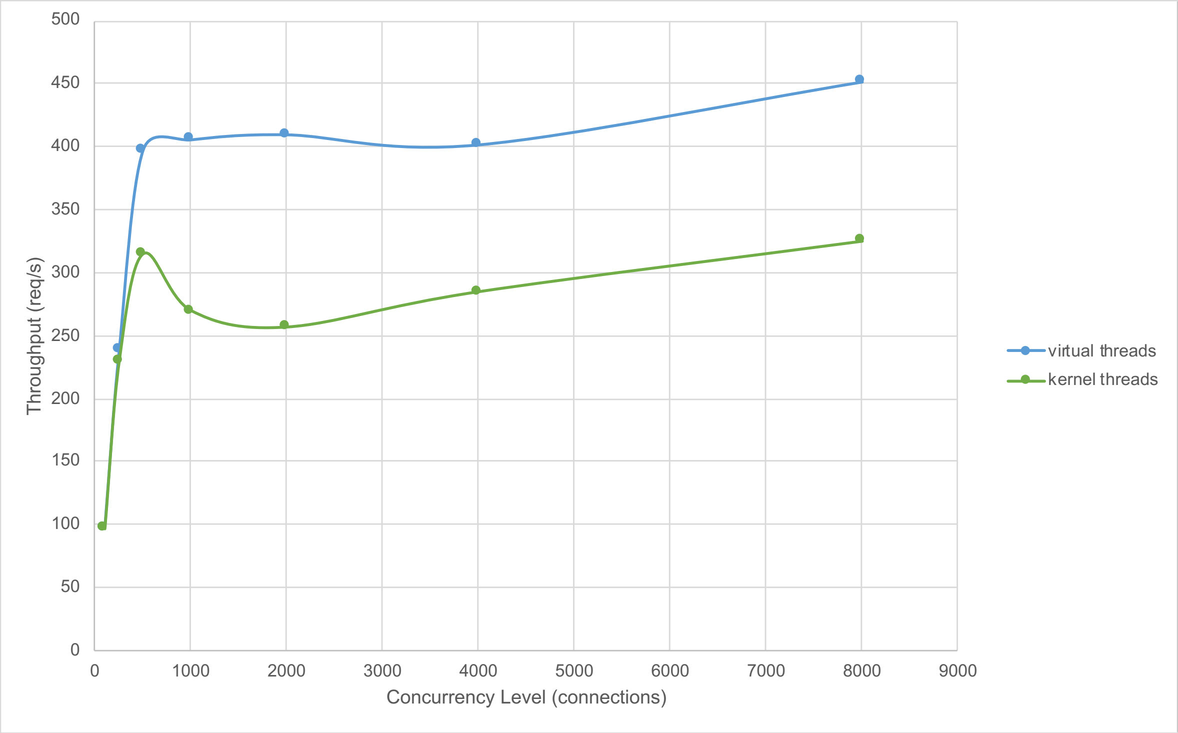


Throughput for 10ms requests



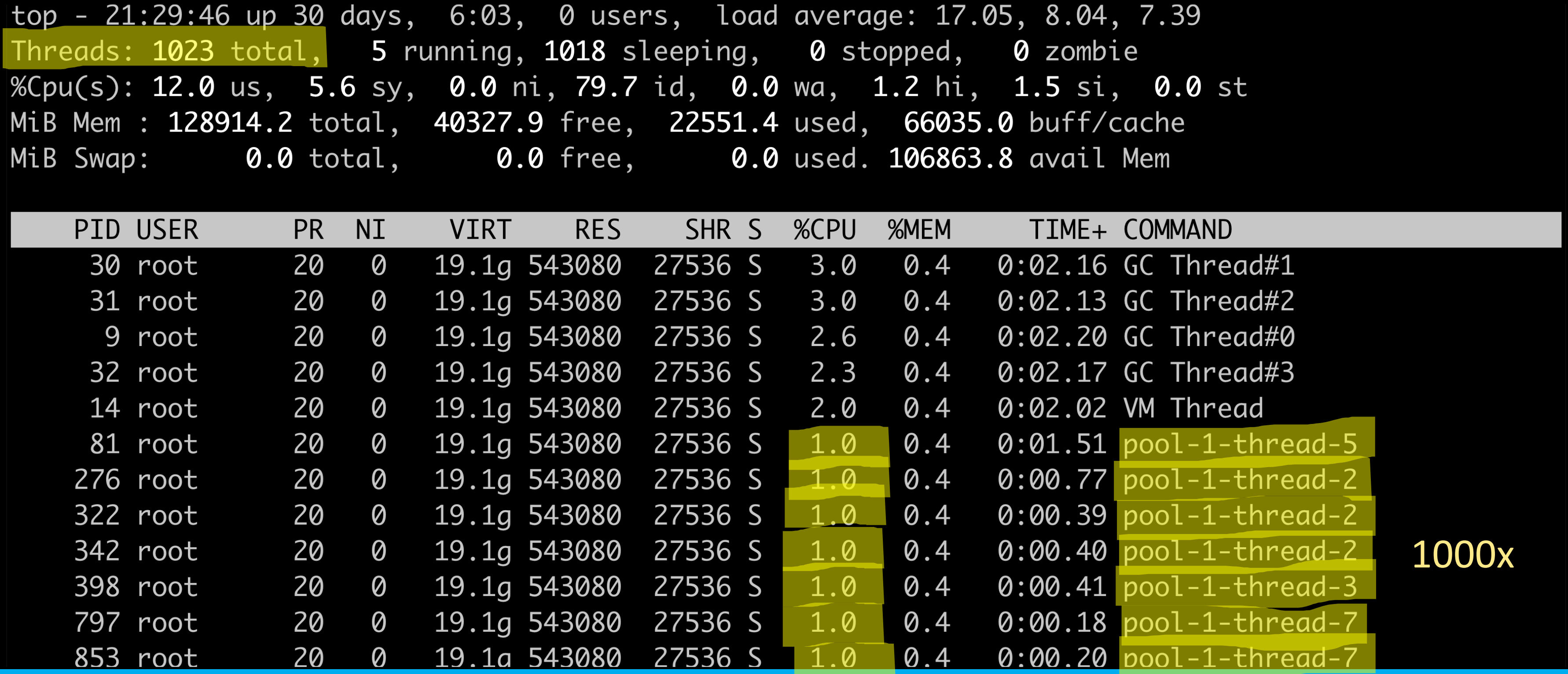
Concurrency	virtual:kernel
100	**99%**
250	**104%**
500	**132%**
1000	**142%**
2000	**138%**
4000	**145%**
8000	**124%**

Throughput for 1s requests



Concurrency	virtual:kernel
100	**101%**
250	**104%**
500	**126%**
1000	**150%**
2000	**160%**
4000	**141%**
8000	**139%**

Kernel threads resources

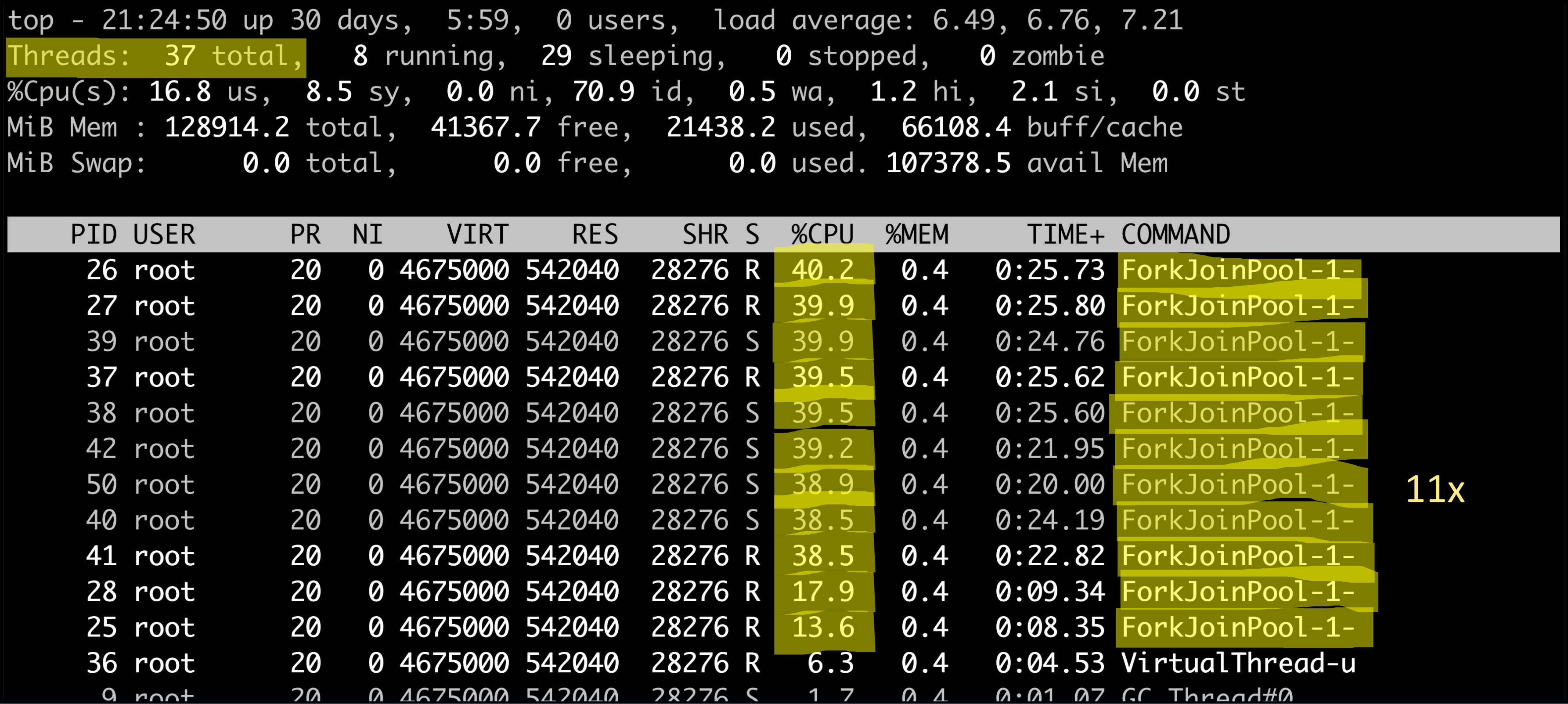


```
top - 21:29:46 up 30 days,  6:03,  0 users,  load average: 17.05, 8.04, 7.39
Threads: 1023 total,   5 running, 1018 sleeping,   0 stopped,   0 zombie
%Cpu(s): 12.0 us,  5.6 sy,  0.0 ni, 79.7 id,  0.0 wa,  1.2 hi,  1.5 si,  0.0 st
MiB Mem : 128914.2 total,  40327.9 free,  22551.4 used,  66035.0 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 106863.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
     30 root      20   0   19.1g 543080  27536 S   3.0   0.4   0:02.16 GC Thread#1
     31 root      20   0   19.1g 543080  27536 S   3.0   0.4   0:02.13 GC Thread#2
      9 root      20   0   19.1g 543080  27536 S   2.6   0.4   0:02.20 GC Thread#0
     32 root      20   0   19.1g 543080  27536 S   2.3   0.4   0:02.17 GC Thread#3
     14 root      20   0   19.1g 543080  27536 S   2.0   0.4   0:02.02 VM Thread
     81 root      20   0   19.1g 543080  27536 S   1.0   0.4   0:01.51 pool-1-thread-5
    276 root      20   0   19.1g 543080  27536 S   1.0   0.4   0:00.77 pool-1-thread-2
    322 root      20   0   19.1g 543080  27536 S   1.0   0.4   0:00.39 pool-1-thread-2
    342 root      20   0   19.1g 543080  27536 S   1.0   0.4   0:00.40 pool-1-thread-2
    398 root      20   0   19.1g 543080  27536 S   1.0   0.4   0:00.41 pool-1-thread-3
    797 root      20   0   19.1g 543080  27536 S   1.0   0.4   0:00.18 pool-1-thread-7
    853 root      20   0   19.1g 543080  27536 S   1.0   0.4   0:00.20 pool-1-thread-7
```

1000x

Virtual threads resources



```
top - 21:24:50 up 30 days,  5:59,  0 users,  load average: 6.49, 6.76, 7.21
Threads:  37 total,   8 running,  29 sleeping,   0 stopped,   0 zombie
%Cpu(s): 16.8 us,  8.5 sy,  0.0 ni, 70.9 id,  0.5 wa,  1.2 hi,  2.1 si,  0.0 st
MiB Mem : 128914.2 total,  41367.7 free,  21438.2 used,  66108.4 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 107378.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
     26 root      20   0 4675000 542040  28276 R  40.2   0.4   0:25.73 ForkJoinPool-1-
     27 root      20   0 4675000 542040  28276 R  39.9   0.4   0:25.80 ForkJoinPool-1-
     39 root      20   0 4675000 542040  28276 S  39.9   0.4   0:24.76 ForkJoinPool-1-
     37 root      20   0 4675000 542040  28276 R  39.5   0.4   0:25.62 ForkJoinPool-1-
     38 root      20   0 4675000 542040  28276 S  39.5   0.4   0:25.60 ForkJoinPool-1-
     42 root      20   0 4675000 542040  28276 S  39.2   0.4   0:21.95 ForkJoinPool-1-
     50 root      20   0 4675000 542040  28276 S  38.9   0.4   0:20.00 ForkJoinPool-1-
     40 root      20   0 4675000 542040  28276 S  38.5   0.4   0:24.19 ForkJoinPool-1-
     41 root      20   0 4675000 542040  28276 R  38.5   0.4   0:22.82 ForkJoinPool-1-
     28 root      20   0 4675000 542040  28276 R  17.9   0.4   0:09.34 ForkJoinPool-1-
     25 root      20   0 4675000 542040  28276 R  13.6   0.4   0:08.35 ForkJoinPool-1-
     36 root      20   0 4675000 542040  28276 R   6.3   0.4   0:04.53 VirtualThread-u
      9 root      20   0 4675000 542040  28276 S   1.7   0.4   0:01.07 GC Thread#0
```

11x

Q & A

References

Sling & Loom

Demo Apache Sling & unbounded pool of virtual threads
https://github.com/tmaret/sling-virtual-thread

Eclipse Jetty based test server for virtual threads and benchmark results
https://github.com/tmaret/fibers-server

Loom

OpenJDK wiki, EA builds, status, code, talks
https://wiki.openjdk.java.net/display/loom/Main

Sling API

https://sling.apache.org/apidocs/sling11/index.html