Linnéuniversitetet
Kalmar Växjö

Linnaeus University

1DV512 - Operating System
Individual Assignment 3

Author: Yuyao Duan
Email: yd222br@student.lnu.se
Course Code:1DV512
Semester: Autumn 2021
Supervisors: Lars Karlsson,
Samuele Giussani, Victor Loby

Table of Contents

1. Introduction

File system plays an important role for operating system. By implementing different strategies, the performance of different filesystem implementations may vary differently [1]. From WAFL, NILFS, LFS to XFS, EXT4, BTRFS [2], the computer scientists spare no effort to innovate new algorithm and data structures for new filesystems to improve operating system's performance. In this assignment, the Linux B-Tree Filesystem (BTRFS) will be examined.

First of all, eight questions regarding a comparison of BTRFS and other related file systems will be discussed. Furthermore, a Java-based programming task in terms of multithreaded service will be implemented that will simulate the execution of multi-threads by the CPU.

2. Task 1

In this section, the research report Rodeh et al. (2013) will be discussed [2]. The aim of this part is to present a deeper understanding of BTRFS (the Linux B-tree File System), then eight related questions will be discussed in the following section.

2.1 Did some of the earlier file systems supported by Linux before BTRFS focus on support for larger volumes of stored data? If yes, which ones?

According to this article, there are three file system earlier than BTRFS which support larger volumes of stored data including: EXT4, XFS, LFS [2]. The following table will summarize the related file systems regarding the task requirement:

System Name	Description
EXT4 (2006)	*"The Fourth Extended Filesystem (EXT4) [Mathur et al. 2007] is a mostly backward compatible extension to the previous general-purpose Linux filesystem, Ext3. It was created to address filesystem and file size limitations, and to improve performance."* [2]
XFS (2000)	*"The design goal of XFS is to achieve high scalability in terms of IO threads, number of disks, file/filesystem size. It is an overwrite class filesystem, which uses B-tree of extents to manage disk space."* [2]
LFS (1990s)	*"The main idea is to write all modifications to disk in large chunks, called segments. The filesystem can be thought of as a large tree that has a well-known root, but whose data is spread across the live segments. When disk space runs low, a segment cleaner background process kicks in. It frees segments by merging several partly full segments, into a smaller number of full segments."* [2]

2.2 Does BTRFS share any ideas and design goals with ZFS? If yes, which ones?

BTRFS and ZFS have several aspects in common: First of all, both BTRFS and ZFS lack of FSCK utilities; for ZFS, it has not such utility; for BTRFS, it is very hard to write a good recovery tool [2]. Moreover, both ZFS and BTRFS calculate checksums, though the implementation strategies are different [2]. Furthermore, BTRFS and ZFS can build snapshots that BTRFS can support clones in addition. Last but not least, both BTRFS and ZFS support RAID that ZFS uses RAID-Z, and supports several RAID levels while BTRFS uses something closer to a standard RAID layout.

2.3 What is the "extent" concept used for storage by BTRFS? What are its pros and cons compared to the alternative approaches?

According to the article, an extent refers to "a contiguous on-disk area. It is page-aligned, and its length is a multiple of pages" [2]. In more detailed, an extent refers to a mapping from a logical area in a file to a physical area on disk [2]. The advantage of using extent includes the following aspects: First, when a file is stored in a small number of large extents, then a common operation such as reading a full file can be efficiently mapped to a few disk operations [2]. Second, using an extent representation can be more space efficient compared to the filesystems using fixed size blocks [2]. The disadvantage of using an extent is also obvious that brings the cost of more complexity [2].

2.4 Which structures does BTRFS use to support larger storage devices and map between physical and logical storage space?

In Linux system, it has two device management subsystems: the *device mapper* (DM) and the *software RAID subsystem* (MD) [2]. However, both modules do not support checksums which causes a problem for BTRFS [2]. To solve the above issue, BTRFS does its own device management by calculating checksums, stores them in a separate tree, and is then better positioned to recover data when media errors occur [2]. There are several tree structures are applied to improve this implementation [2]. The structure *chunk tree* is used to maintain a mapping from logical chunks to physical chunks; a *device tree* maintains the reverse mapping [2]. The rest of the filesystem sees logical chunks, and all extent references address logical chunks [2]. This allows moving physical chunks under the covers without the need to back trace and fix references [2]. The chunk/device trees are small, and can typically be cached in memory which could reduce the performance cost of an added indirection layer [2]. Physical chunks are divided into groups according to the required RAID level of the logical chunk [2]. For mirroring, chunks are divided into pairs [2].

2.5 Does BTRFS support copy-on-write? If yes, is it possible to disable it? Could such an approach have impact on performance and fragmentation?

BTRFS supports copy-on-write, the *nocow* option can be used to cancel copy-on-right for data blocks, unless there is a snapshot [2]. According to the article, even NOCOW makes sense for workloads.

However, COW could be very expensive which is due to the database workloads could do random small updates, and then followed by sequential scans [2]. By using BTRFS, it will write the blocks to disk in update order, which will lead to very bad performance in the sequential scan [2]. Ultimately, databases could well be large enough to overwhelm the in-memory buffers, defeating the attempt to lay out the data in increasing address order [2].

2.6 Do changes to the file system immediately get written to the disk, when using BTRFS? How does this proceed?

In BTRFS filesystem, the changes do not immediately get written to the disk. The modifications are firstly accumulated in memory. Then after a timeout, or enough pages have been changed, the changes will be written in batch to new disk locations that will form a checkpoint [2]. Once the checkpoint has been written, the superblock is modified to point to the new checkpoint; this is the only case where a disk block is modified in place. In case crash occurs, the filesystem recovers by reading the superblock, and following the pointers to the last valid on-disk checkpoint [2]. When a checkpoint is initiated, all dirty memory pages that are part of it are marked immutable [2]. The updates of the user will be received while the checkpoint is in flight cause immutable pages to be re-COWed [2]. This allows user visible filesystem operations to proceed without damaging checkpoint integrity [2]. In this filesystem, the checkpoints are numbered by increasing generation number that is embedded in various on disk data structure [2].

2.7 Did Rodeh et al. discover large differences in performance between BTRFS and ZFS in their comparisons? How/why?

In this research, Rodeh et al. did not compare the performance between BTRFS and ZFS. The performance comparison needs to conduct under Linux system environment [2]. According to the article, the features of ZFS have significant differences compared to BTRFS. Even ZFS has ports to Linux, it is not native Linux system, therefore it prevents a reliable comparison between ZFS and BTRFS [2].

2.8 Which workloads did Rodeh et al. use for their benchmark tests with FileBench? How did BTRFS compare to the more mature file systems in these tests?

In this research, Rodeh et al. used four main types of workloads: web, file, mail, OLTP [2]. The summary of detail information can be found as following:

Workload	Avg. file size	#files	Footprint	IO size (R/W)	#threads	R/W ratio
web	32KB	350000	11GB	1MB/16KB	100	10:1
file	256KB	50000	12.5GB	1MB/16KB	50	1:2
mail	16KB	800000	12.5GB	1MB/16KB	100	1:1

OLTP	1GB	10	10GB	2KB/2KB	200+10	20:1

In this research, the Web workload emulates a Web server that serves files to HTTP clients; the file workload mimics a server that hosts home directories of multiple users; the mail workload mimics an electronic mail server which creates a flat directory structure with many small files; the OLTP workload emulates a database that performs online transaction processing that is based on the IO model used by Oracle TM9i [2]. To measure these workloads, this research used the operations per second (*ops/s*), and the cpu per operation (*cpu/op*) metrics to inspect the results [2]. An *operation* is a benchmark step, such as reading a whole file, appending a certain amount of data to it, and so on [2]. Ops/s can be used to compare relative filesystem performance [2].

In this experiment, Rodeh et al. used two types of storage devices to test the result: hard disk and SSD. Two types of devices show shed lights on different insights. Generally speaking, BTRFS requires more CPU cycles in terms of cpu/op values for both HDD and SSD [2]. By using HDD, BTRFS has similar performance for file and OLTP workloads, while it is about four to eight times slower than Ext4 in terms of the mail workload (Ext4 using HTree data structure is a very effective index for the large flat mail directory) and two times slower than Ext4 regarding web workload (B-tree indexes used by the other filesystems are much deeper, requiring more random IO which leads to Ext4 having more advantage) [2].

By using SSD, the differences between BTRFS and the other filesystems are less significant in this research. The ops/s shows that all three filesystems are reasonably close for the file and Web workloads [2], which we can find that BTRFS has a slight lead in file workload. BTRFS is much less efficient in the mail workload compared to other two filesystems which is due to its less efficient fsync implementation. With respect to OLTP, BTRFS performed poorly with copy-on-write, this disadvantage becomes not obvious until disabling COW for BTRFS [2].

3. Task 2

In this section, the experiments regarding the implementation of a multithreaded algorithm based on Java programming language will be presented. The aim of this part is to explore the features of the multithreaded CPU processing.

3.1 General comments of the Multithreaded Service in Java

In this implementation, a multithreaded algorithm-based Java program is created. The first important part of this program is Task.java class which extends Thread.java class in order to prepare for overriding run method. ArrayList is used as the container to collect all the task objects. The main logic of “runNewSimulation()” method is as follows: The system start time and end time will be recorded. Then, the ExecutorService object will be created for creating fixed number of threads

required in the task. The required number of tasks will be created and added in the list. A task is considered as finished when allocated CPU time greater or equal to burst time then set the current system to the endTime and submit the task to run. A while loop is used to simulate the 15 seconds program running. If the system times out, the ExecutorService object will shut down immediately. In the printResults() method, the SimpleDateFormat object is used to format system milliseconds to a meaningful expression.

3.2 Simulation results

3.2.1 Run Simulation #0

```
Running simulation #0
Simulation results:
---------------------
Completed tasks:
Task ID      Burst Time    Start Time(Hour:Min:Sec:Ms)       Finish Time(Hour:Min:Sec:Ms)
    0            8957              19:33:36:126                      19:33:45:382
    1            1206              19:33:36:126                      19:33:37:446
    2            8303              19:33:36:127                      19:33:44:787
    3            9187              19:33:36:127                      19:33:45:595
    4            9452              19:33:37:448                      19:33:47:230
    5            2124              19:33:44:788                      19:33:47:057
    7            4301              19:33:45:595                      19:33:50:120
Interrupted tasks:
Task ID      Burst Time    Start Time(Hour:Min:Sec:Ms)
    6            5782              19:33:45:383
    8            8452              19:33:47:057
    9            4655              19:33:47:230
   10            2660              19:33:50:120
Waiting tasks:
Task ID      Burst Time
   11            8171
   12            1091
   13            5050
   14            6527
   15            9072
   16            2790
   17            6903
   18            5877
   19            8930
   20            5461
   21            7036
   22            4497
   23            8045
   24            3202
   25            4375
   26            9966
   27            4968
   28            6801
   29            2394
```

3.2.2 Run Simulation #1

```
Running simulation #1
Simulation results:
---------------------
Completed tasks:
Task ID      Burst Time    Start Time(Hour:Min:Sec:Ms)       Finish Time(Hour:Min:Sec:Ms)
    0            3952              19:33:51:175                      19:33:55:289
    1            6643              19:33:51:175                      19:33:58:044
    2            3616              19:33:51:175                      19:33:54:971
    3            3782              19:33:51:175                      19:33:55:088
    4            6933              19:33:54:971                      19:34:02:169
    5            8309              19:33:55:088                      19:34:03:742
    6            3376              19:33:55:289                      19:33:58:782
    8            2745              19:33:58:782                      19:34:01:661
Interrupted tasks:
Task ID      Burst Time    Start Time(Hour:Min:Sec:Ms)
    7            9172              19:33:58:044
    9            9362              19:34:01:661
   10            5642              19:34:02:169
   11            5335              19:34:03:742
Waiting tasks:
Task ID      Burst Time
   12            6840
   13            1824
   14            9998
   15            1007
   16            3215
   17            5834
   18            5045
   19            4226
   20            5528
   21            5430
   22            5469
   23            8996
   24            4582
   25            7160
   26            2922
   27            2057
   28            8358
   29            7965
```

3.2.3 Run Simulation #2

```
Running simulation #2
Simulation results:
---------------------
Completed tasks:
Task ID      Burst Time    Start Time(Hour:Min:Sec:Ms)        Finish Time(Hour:Min:Sec:Ms)
    0            1746              19:34:06:180                       19:34:08:023
    1            5411              19:34:06:180                       19:34:11:805
    2            3026              19:34:06:181                       19:34:09:345
    3            9359              19:34:06:181                       19:34:15:835
    4            3977              19:34:08:023                       19:34:12:107
    5            2980              19:34:09:345                       19:34:12:415
    6            5258              19:34:11:805                       19:34:17:240
    7            8632              19:34:12:108                       19:34:21:077
Interrupted tasks:
Task ID      Burst Time    Start Time(Hour:Min:Sec:Ms)
    8            9638              19:34:12:415
    9            7301              19:34:15:835
   10            9037              19:34:17:240
   11            6273              19:34:21:077
Waiting tasks:
Task ID      Burst Time
   12            3493
   13            4211
   14            2821
   15            7922
   16            9766
   17            8177
   18            5309
   19            4029
   20            2814
   21            6531
   22            4723
   23            2751
   24            4780
   25            3919
   26            3809
   27            8872
   28            3104
   29            2237



---------------------
Exiting...
```

3.3 Acquiring process ID and thread dump

3.3.1 Using jps command to acquire process ID

```
(base) YMacBook:~ yuyaoduan$ jps -l
7793 org.jetbrains.jps.cmdline.Launcher
7794 MultithreadedService
7795 jdk.jcmd/sun.tools.jps.Jps
516
```

3.3.2 Using jstack to acquire thread dump

3.3.2.1 Thread dump of simulation #0

```
(base) YMacBook:~ yuyaoduan$ jstack -l 7794
2021-12-26 20:02:57
Full thread dump OpenJDK 64-Bit Server VM (17.0.1+12-39 mixed mode, sharing):

Threads class SMR info:
_java_thread_list=0x00007ff4f8f0b5a0, length=17, elements={
0x00007ff4f9800000, 0x00007ff4f982ce00, 0x00007ff4f9000600, 0x00007ff4fa83fa00,
0x00007ff4fa006800, 0x00007ff4fa006e00, 0x00007ff4fa008400, 0x00007ff4f9001200,
0x00007ff4fa840000, 0x00007ff4f9046a00, 0x00007ff4f9045c00, 0x00007ff4f904a800,
0x00007ff4f9805e00, 0x00007ff4f908f600, 0x00007ff4fa008a00, 0x00007ff4fa00c200,
0x00007ff4f9806400
}
```

The other information of this dump can be found in the following table:

```
"main" #1 prio=5 os_prio=31 cpu=9554.51ms elapsed=10.32s tid=0x00007ff4f9800000
nid=0x2103 runnable  [0x0000700001444000]
   java.lang.Thread.State: RUNNABLE
        at MultithreadedService.runNewSimulation(MultithreadedService.java:136)
        at MultithreadedService.main(MultithreadedService.java:228)

   Locked ownable synchronizers:
        - None

"Reference Handler" #2 daemon prio=10 os_prio=31 cpu=0.19ms elapsed=10.30s
tid=0x00007ff4f982ce00 nid=0x4203 waiting on condition  [0x0000700001b5a000]
   java.lang.Thread.State: RUNNABLE
        at java.lang.ref.Reference.waitForReferencePendingList(java.base@17.0.1/Native
Method)
        at
java.lang.ref.Reference.processPendingReferences(java.base@17.0.1/Reference.java:253)
        at java.lang.ref.Reference$ReferenceHandler.run(java.base@17.0.1/Reference.java:215)
```

```
   Locked ownable synchronizers:
        - None

"Finalizer" #3 daemon prio=8 os_prio=31 cpu=0.32ms elapsed=10.30s tid=0x00007ff4f9000600
nid=0x5503 in Object.wait()  [0x0000700001c5d000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(java.base@17.0.1/Native Method)
        - waiting on <0x0000000787f02f30> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:155)
        - locked <0x0000000787f02f30> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:176)
        at java.lang.ref.Finalizer$FinalizerThread.run(java.base@17.0.1/Finalizer.java:172)

   Locked ownable synchronizers:
        - None

"Signal Dispatcher" #4 daemon prio=9 os_prio=31 cpu=0.49ms elapsed=10.28s
tid=0x00007ff4fa83fa00 nid=0xa503 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Service Thread" #5 daemon prio=9 os_prio=31 cpu=0.13ms elapsed=10.28s
tid=0x00007ff4fa006800 nid=0xa303 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Monitor Deflation Thread" #6 daemon prio=9 os_prio=31 cpu=0.21ms elapsed=10.28s
tid=0x00007ff4fa006e00 nid=0xa203 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
```

```
   Locked ownable synchronizers:
        - None

"C2 CompilerThread0" #7 daemon prio=9 os_prio=31 cpu=94.46ms elapsed=10.28s
tid=0x00007ff4fa008400 nid=0x5c03 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"C1 CompilerThread0" #9 daemon prio=9 os_prio=31 cpu=94.11ms elapsed=10.28s
tid=0x00007ff4f9001200 nid=0x5e03 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"Sweeper thread" #10 daemon prio=9 os_prio=31 cpu=0.09ms elapsed=10.27s
tid=0x00007ff4fa840000 nid=0x5f03 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Common-Cleaner" #11 daemon prio=8 os_prio=31 cpu=0.20ms elapsed=10.23s
tid=0x00007ff4f9046a00 nid=0xa003 in Object.wait()  [0x0000700002372000]
   java.lang.Thread.State: TIMED_WAITING (on object monitor)
        at java.lang.Object.wait(java.base@17.0.1/Native Method)
        - waiting on <0x000000787f424f8> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:155)
        - locked <0x000000787f424f8> (a java.lang.ref.ReferenceQueue$Lock)
```

```
        at jdk.internal.ref.CleanerImpl.run(java.base@17.0.1/CleanerImpl.java:140)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)
        at jdk.internal.misc.InnocuousThread.run(java.base@17.0.1/InnocuousThread.java:162)

   Locked ownable synchronizers:
        - None

"Monitor Ctrl-Break" #12 daemon prio=5 os_prio=31 cpu=65.20ms elapsed=10.07s
tid=0x00007ff4f9045c00 nid=0x9f03 runnable  [0x0000700002475000]
   java.lang.Thread.State: RUNNABLE
        at sun.nio.ch.SocketDispatcher.read0(java.base@17.0.1/Native Method)
        at sun.nio.ch.SocketDispatcher.read(java.base@17.0.1/SocketDispatcher.java:47)
        at sun.nio.ch.NioSocketImpl.tryRead(java.base@17.0.1/NioSocketImpl.java:261)
        at sun.nio.ch.NioSocketImpl.implRead(java.base@17.0.1/NioSocketImpl.java:312)
        at sun.nio.ch.NioSocketImpl.read(java.base@17.0.1/NioSocketImpl.java:350)
        at sun.nio.ch.NioSocketImpl$1.read(java.base@17.0.1/NioSocketImpl.java:803)
        at java.net.Socket$SocketInputStream.read(java.base@17.0.1/Socket.java:966)
        at sun.nio.cs.StreamDecoder.readBytes(java.base@17.0.1/StreamDecoder.java:270)
        at sun.nio.cs.StreamDecoder.implRead(java.base@17.0.1/StreamDecoder.java:313)
        at sun.nio.cs.StreamDecoder.read(java.base@17.0.1/StreamDecoder.java:188)
        - locked <0x0000000787e9e100> (a java.io.InputStreamReader)
        at java.io.InputStreamReader.read(java.base@17.0.1/InputStreamReader.java:177)
        at java.io.BufferedReader.fill(java.base@17.0.1/BufferedReader.java:162)
        at java.io.BufferedReader.readLine(java.base@17.0.1/BufferedReader.java:329)
        - locked <0x0000000787e9e100> (a java.io.InputStreamReader)
        at java.io.BufferedReader.readLine(java.base@17.0.1/BufferedReader.java:396)
        at com.intellij.rt.execution.application.AppMainV2$1.run(AppMainV2.java:49)

   Locked ownable synchronizers:
        - <0x0000000787e93d20> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)

"Notification Thread" #13 daemon prio=9 os_prio=31 cpu=0.07ms elapsed=10.07s
tid=0x00007ff4f904a800 nid=0x9e03 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
```

```
   Locked ownable synchronizers:
        - None

"pool-1-thread-1" #15 prio=5 os_prio=31 cpu=3.33ms elapsed=10.01s tid=0x00007ff4f9805e00
nid=0x9c03 waiting on condition  [0x000070000277e000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787edb900> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-1-thread-2" #17 prio=5 os_prio=31 cpu=3.35ms elapsed=10.01s tid=0x00007ff4f908f600
nid=0x9b03 waiting on condition  [0x0000700002881000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
```

```
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787edc208> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-1-thread-3" #19 prio=5 os_prio=31 cpu=2.48ms elapsed=10.01s tid=0x00007ff4fa008a00
nid=0x6603 waiting on condition  [0x0000700002984000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787edc7a0> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-1-thread-4" #21 prio=5 os_prio=31 cpu=2.46ms elapsed=10.01s tid=0x00007ff4fa00c200
nid=0x9803 waiting on condition  [0x0000700002a87000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
```

```
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787edcd68> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"Attach Listener" #48 daemon prio=9 os_prio=31 cpu=38.37ms elapsed=9.15s
tid=0x00007ff4f9806400 nid=0x6803 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"VM Thread" os_prio=31 cpu=3.36ms elapsed=10.30s tid=0x00007ff4f8d0eee0 nid=0x3f03
runnable

"GC Thread#0" os_prio=31 cpu=0.21ms elapsed=10.32s tid=0x00007ff4f8d02970 nid=0x3403
runnable

"G1 Main Marker" os_prio=31 cpu=0.06ms elapsed=10.32s tid=0x00007ff4f8d037c0 nid=0x3503
runnable

"G1 Conc#0" os_prio=31 cpu=0.04ms elapsed=10.32s tid=0x00007ff4f8d04520 nid=0x3703
runnable

"G1 Refine#0" os_prio=31 cpu=0.08ms elapsed=10.32s tid=0x00007ff4f8c27250 nid=0x4803
runnable
```

"G1 Service" os_prio=31 cpu=1.23ms elapsed=10.32s tid=0x00007ff4f8f01190 nid=0x4603 runnable

"VM Periodic Task Thread" os_prio=31 cpu=3.47ms elapsed=10.07s tid=0x00007ff4f8c58af0 nid=0x6403 waiting on condition

JNI global refs: 14, weak refs: 0

3.3.2.2 Thread dump of simulation #1

```
(base) YMacBook:~ yuyaoduan$ jstack -l 7794
2021-12-26 20:03:08
Full thread dump OpenJDK 64-Bit Server VM (17.0.1+12-39 mixed mode, sharing):

Threads class SMR info:
_java_thread_list=0x00007ff4f8d2d8a0, length=17, elements={
0x00007ff4f9800000, 0x00007ff4f982ce00, 0x00007ff4f9000600, 0x00007ff4fa83fa00,
0x00007ff4fa006800, 0x00007ff4fa006e00, 0x00007ff4fa008400, 0x00007ff4f9001200,
0x00007ff4fa840000, 0x00007ff4f9046a00, 0x00007ff4f9045c00, 0x00007ff4f904a800,
0x00007ff4f9806400, 0x00007ff4fa008a00, 0x00007ff4fa04ca00, 0x00007ff4fa04d000,
0x00007ff4fa04d600
}
```

The other information of this dump can be found in the following table:

"main" #1 prio=5 os_prio=31 cpu=20347.59ms elapsed=21.33s tid=0x00007ff4f9800000 nid=0x2103 runnable [0x0000700001444000]
   java.lang.Thread.State: RUNNABLE
        at MultithreadedService.runNewSimulation(MultithreadedService.java:136)
        at MultithreadedService.main(MultithreadedService.java:228)

   Locked ownable synchronizers:
        - None

"Reference Handler" #2 daemon prio=10 os_prio=31 cpu=0.19ms elapsed=21.31s tid=0x00007ff4f982ce00 nid=0x4203 waiting on condition [0x0000700001b5a000]
   java.lang.Thread.State: RUNNABLE
        at java.lang.ref.Reference.waitForReferencePendingList(java.base@17.0.1/Native Method)
        at

```
java.lang.ref.Reference.processPendingReferences(java.base@17.0.1/Reference.java:253)
        at java.lang.ref.Reference$ReferenceHandler.run(java.base@17.0.1/Reference.java:215)

   Locked ownable synchronizers:
        - None

"Finalizer" #3 daemon prio=8 os_prio=31 cpu=0.32ms elapsed=21.31s tid=0x00007ff4f9000600
nid=0x5503 in Object.wait()  [0x0000700001c5d000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(java.base@17.0.1/Native Method)
        - waiting on <0x0000000787f02f30> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:155)
        - locked <0x0000000787f02f30> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:176)
        at java.lang.ref.Finalizer$FinalizerThread.run(java.base@17.0.1/Finalizer.java:172)

   Locked ownable synchronizers:
        - None

"Signal Dispatcher" #4 daemon prio=9 os_prio=31 cpu=0.49ms elapsed=21.29s
tid=0x00007ff4fa83fa00 nid=0xa503 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Service Thread" #5 daemon prio=9 os_prio=31 cpu=0.21ms elapsed=21.29s
tid=0x00007ff4fa006800 nid=0xa303 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Monitor Deflation Thread" #6 daemon prio=9 os_prio=31 cpu=0.38ms elapsed=21.29s
```

```
tid=0x00007ff4fa006e00 nid=0xa203 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"C2 CompilerThread0" #7 daemon prio=9 os_prio=31 cpu=182.24ms elapsed=21.29s
tid=0x00007ff4fa008400 nid=0x5c03 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"C1 CompilerThread0" #9 daemon prio=9 os_prio=31 cpu=142.26ms elapsed=21.29s
tid=0x00007ff4f9001200 nid=0x5e03 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"Sweeper thread" #10 daemon prio=9 os_prio=31 cpu=0.09ms elapsed=21.28s
tid=0x00007ff4fa840000 nid=0x5f03 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Common-Cleaner" #11 daemon prio=8 os_prio=31 cpu=0.20ms elapsed=21.24s
tid=0x00007ff4f9046a00 nid=0xa003 in Object.wait()  [0x0000700002372000]
   java.lang.Thread.State: TIMED_WAITING (on object monitor)
        at java.lang.Object.wait(java.base@17.0.1/Native Method)
        - waiting on <0x000000787f424f8> (a java.lang.ref.ReferenceQueue$Lock)
```

```
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:155)
        - locked <0x0000000787f424f8> (a java.lang.ref.ReferenceQueue$Lock)
        at jdk.internal.ref.CleanerImpl.run(java.base@17.0.1/CleanerImpl.java:140)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)
        at jdk.internal.misc.InnocuousThread.run(java.base@17.0.1/InnocuousThread.java:162)

   Locked ownable synchronizers:
        - None

"Monitor Ctrl-Break" #12 daemon prio=5 os_prio=31 cpu=65.20ms elapsed=21.08s
tid=0x00007ff4f9045c00 nid=0x9f03 runnable  [0x0000700002475000]
   java.lang.Thread.State: RUNNABLE
        at sun.nio.ch.SocketDispatcher.read0(java.base@17.0.1/Native Method)
        at sun.nio.ch.SocketDispatcher.read(java.base@17.0.1/SocketDispatcher.java:47)
        at sun.nio.ch.NioSocketImpl.tryRead(java.base@17.0.1/NioSocketImpl.java:261)
        at sun.nio.ch.NioSocketImpl.implRead(java.base@17.0.1/NioSocketImpl.java:312)
        at sun.nio.ch.NioSocketImpl.read(java.base@17.0.1/NioSocketImpl.java:350)
        at sun.nio.ch.NioSocketImpl$1.read(java.base@17.0.1/NioSocketImpl.java:803)
        at java.net.Socket$SocketInputStream.read(java.base@17.0.1/Socket.java:966)
        at sun.nio.cs.StreamDecoder.readBytes(java.base@17.0.1/StreamDecoder.java:270)
        at sun.nio.cs.StreamDecoder.implRead(java.base@17.0.1/StreamDecoder.java:313)
        at sun.nio.cs.StreamDecoder.read(java.base@17.0.1/StreamDecoder.java:188)
        - locked <0x0000000787e9e100> (a java.io.InputStreamReader)
        at java.io.InputStreamReader.read(java.base@17.0.1/InputStreamReader.java:177)
        at java.io.BufferedReader.fill(java.base@17.0.1/BufferedReader.java:162)
        at java.io.BufferedReader.readLine(java.base@17.0.1/BufferedReader.java:329)
        - locked <0x0000000787e9e100> (a java.io.InputStreamReader)
        at java.io.BufferedReader.readLine(java.base@17.0.1/BufferedReader.java:396)
        at com.intellij.rt.execution.application.AppMainV2$1.run(AppMainV2.java:49)

   Locked ownable synchronizers:
        - <0x0000000787e93d20> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)

"Notification Thread" #13 daemon prio=9 os_prio=31 cpu=0.07ms elapsed=21.08s
```

```
tid=0x00007ff4f904a800 nid=0x9e03 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Attach Listener" #48 daemon prio=9 os_prio=31 cpu=38.85ms elapsed=20.17s
tid=0x00007ff4f9806400 nid=0x6803 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"pool-2-thread-1" #50 prio=5 os_prio=31 cpu=1.31ms elapsed=5.99s tid=0x00007ff4fa008a00
nid=0x6607 waiting on condition  [0x000070000277e000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787bfc158> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-2-thread-2" #52 prio=5 os_prio=31 cpu=1.29ms elapsed=5.99s tid=0x00007ff4fa04ca00
nid=0x9907 waiting on condition  [0x0000700002881000]
```

```
  java.lang.Thread.State: TIMED_WAITING (sleeping)
       at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
       at MultithreadedService$1.run(MultithreadedService.java:116)
       at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
       at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
       at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
       at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
       at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

  Locked ownable synchronizers:
       - <0x0000000787bfc748> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-2-thread-3" #54 prio=5 os_prio=31 cpu=1.49ms elapsed=5.99s tid=0x00007ff4fa04d000
nid=0x9b07 waiting on condition  [0x0000700002984000]
  java.lang.Thread.State: TIMED_WAITING (sleeping)
       at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
       at MultithreadedService$1.run(MultithreadedService.java:116)
       at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
       at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
       at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
       at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
       at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

  Locked ownable synchronizers:
```

```
        - <0x0000000787bfcce8> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-2-thread-4" #56 prio=5 os_prio=31 cpu=1.35ms elapsed=5.99s tid=0x00007ff4fa04d600
nid=0x9c07 waiting on condition  [0x0000700002a87000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787bfd288> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"VM Thread" os_prio=31 cpu=6.04ms elapsed=21.31s tid=0x00007ff4f8d0eee0 nid=0x3f03
runnable

"GC Thread#0" os_prio=31 cpu=0.28ms elapsed=21.33s tid=0x00007ff4f8d02970 nid=0x3403
runnable

"G1 Main Marker" os_prio=31 cpu=0.06ms elapsed=21.33s tid=0x00007ff4f8d037c0 nid=0x3503
runnable

"G1 Conc#0" os_prio=31 cpu=0.04ms elapsed=21.33s tid=0x00007ff4f8d04520 nid=0x3703
runnable

"G1 Refine#0" os_prio=31 cpu=0.08ms elapsed=21.33s tid=0x00007ff4f8c27250 nid=0x4803
```

runnable

"G1 Service" os_prio=31 cpu=2.25ms elapsed=21.33s tid=0x00007ff4f8f01190 nid=0x4603 runnable

"VM Periodic Task Thread" os_prio=31 cpu=6.99ms elapsed=21.08s tid=0x00007ff4f8c58af0 nid=0x6403 waiting on condition

JNI global refs: 14, weak refs: 0

3.3.2.3 Thread dump of simulation #2

```
(base) YMacBook:~ yuyaoduan$ jstack -l 7794
2021-12-26 20:03:22
Full thread dump OpenJDK 64-Bit Server VM (17.0.1+12-39 mixed mode, sharing):

Threads class SMR info:
_java_thread_list=0x00007ff4f8d2dae0, length=17, elements={
0x00007ff4f9800000, 0x00007ff4f982ce00, 0x00007ff4f9000600, 0x00007ff4fa83fa00,
0x00007ff4fa006800, 0x00007ff4fa006e00, 0x00007ff4fa008400, 0x00007ff4f9001200,
0x00007ff4fa840000, 0x00007ff4f9046a00, 0x00007ff4f9045c00, 0x00007ff4f904a800,
0x00007ff4f9806400, 0x00007ff4f908a000, 0x00007ff4f908a600, 0x00007ff4f9830200,
0x00007ff4f908d000
}
```

The other information of this dump can be found in the following table:

"main" #1 prio=5 os_prio=31 cpu=34360.54ms elapsed=35.59s tid=0x00007ff4f9800000 nid=0x2103 runnable [0x0000700001444000]
  java.lang.Thread.State: RUNNABLE
    at MultithreadedService.runNewSimulation(MultithreadedService.java:136)
    at MultithreadedService.main(MultithreadedService.java:228)

  Locked ownable synchronizers:
    - None

"Reference Handler" #2 daemon prio=10 os_prio=31 cpu=0.19ms elapsed=35.57s tid=0x00007ff4f982ce00 nid=0x4203 waiting on condition [0x0000700001b5a000]
  java.lang.Thread.State: RUNNABLE
    at java.lang.ref.Reference.waitForReferencePendingList(java.base@17.0.1/Native

```
Method)
        at
java.lang.ref.Reference.processPendingReferences(java.base@17.0.1/Reference.java:253)
        at java.lang.ref.Reference$ReferenceHandler.run(java.base@17.0.1/Reference.java:215)

   Locked ownable synchronizers:
        - None

"Finalizer" #3 daemon prio=8 os_prio=31 cpu=0.32ms elapsed=35.57s tid=0x00007ff4f9000600
nid=0x5503 in Object.wait()  [0x0000700001c5d000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(java.base@17.0.1/Native Method)
        - waiting on <0x0000000787f02f30> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:155)
        - locked <0x0000000787f02f30> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:176)
        at java.lang.ref.Finalizer$FinalizerThread.run(java.base@17.0.1/Finalizer.java:172)

   Locked ownable synchronizers:
        - None

"Signal Dispatcher" #4 daemon prio=9 os_prio=31 cpu=0.49ms elapsed=35.55s
tid=0x00007ff4fa83fa00 nid=0xa503 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Service Thread" #5 daemon prio=9 os_prio=31 cpu=0.24ms elapsed=35.55s
tid=0x00007ff4fa006800 nid=0xa303 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None
```

```
"Monitor Deflation Thread" #6 daemon prio=9 os_prio=31 cpu=0.58ms elapsed=35.55s
tid=0x00007ff4fa006e00 nid=0xa203 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"C2 CompilerThread0" #7 daemon prio=9 os_prio=31 cpu=187.62ms elapsed=35.55s
tid=0x00007ff4fa008400 nid=0x5c03 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"C1 CompilerThread0" #9 daemon prio=9 os_prio=31 cpu=146.53ms elapsed=35.55s
tid=0x00007ff4f9001200 nid=0x5e03 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"Sweeper thread" #10 daemon prio=9 os_prio=31 cpu=0.09ms elapsed=35.54s
tid=0x00007ff4fa840000 nid=0x5f03 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Common-Cleaner" #11 daemon prio=8 os_prio=31 cpu=0.20ms elapsed=35.50s
tid=0x00007ff4f9046a00 nid=0xa003 in Object.wait()  [0x0000700002372000]
   java.lang.Thread.State: TIMED_WAITING (on object monitor)
```

```
        at java.lang.Object.wait(java.base@17.0.1/Native Method)
        - waiting on <0x0000000787f424f8> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:155)
        - locked <0x0000000787f424f8> (a java.lang.ref.ReferenceQueue$Lock)
        at jdk.internal.ref.CleanerImpl.run(java.base@17.0.1/CleanerImpl.java:140)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)
        at jdk.internal.misc.InnocuousThread.run(java.base@17.0.1/InnocuousThread.java:162)

   Locked ownable synchronizers:
        - None

"Monitor Ctrl-Break" #12 daemon prio=5 os_prio=31 cpu=65.20ms elapsed=35.34s
tid=0x00007ff4f9045c00 nid=0x9f03 runnable  [0x0000700002475000]
   java.lang.Thread.State: RUNNABLE
        at sun.nio.ch.SocketDispatcher.read0(java.base@17.0.1/Native Method)
        at sun.nio.ch.SocketDispatcher.read(java.base@17.0.1/SocketDispatcher.java:47)
        at sun.nio.ch.NioSocketImpl.tryRead(java.base@17.0.1/NioSocketImpl.java:261)
        at sun.nio.ch.NioSocketImpl.implRead(java.base@17.0.1/NioSocketImpl.java:312)
        at sun.nio.ch.NioSocketImpl.read(java.base@17.0.1/NioSocketImpl.java:350)
        at sun.nio.ch.NioSocketImpl$1.read(java.base@17.0.1/NioSocketImpl.java:803)
        at java.net.Socket$SocketInputStream.read(java.base@17.0.1/Socket.java:966)
        at sun.nio.cs.StreamDecoder.readBytes(java.base@17.0.1/StreamDecoder.java:270)
        at sun.nio.cs.StreamDecoder.implRead(java.base@17.0.1/StreamDecoder.java:313)
        at sun.nio.cs.StreamDecoder.read(java.base@17.0.1/StreamDecoder.java:188)
        - locked <0x0000000787e9e100> (a java.io.InputStreamReader)
        at java.io.InputStreamReader.read(java.base@17.0.1/InputStreamReader.java:177)
        at java.io.BufferedReader.fill(java.base@17.0.1/BufferedReader.java:162)
        at java.io.BufferedReader.readLine(java.base@17.0.1/BufferedReader.java:329)
        - locked <0x0000000787e9e100> (a java.io.InputStreamReader)
        at java.io.BufferedReader.readLine(java.base@17.0.1/BufferedReader.java:396)
        at com.intellij.rt.execution.application.AppMainV2$1.run(AppMainV2.java:49)

   Locked ownable synchronizers:
        - <0x0000000787e93d20> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)
```

```
"Notification Thread" #13 daemon prio=9 os_prio=31 cpu=0.07ms elapsed=35.34s
tid=0x00007ff4f904a800 nid=0x9e03 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Attach Listener" #48 daemon prio=9 os_prio=31 cpu=39.56ms elapsed=34.43s
tid=0x00007ff4f9806400 nid=0x6803 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"pool-3-thread-1" #84 prio=5 os_prio=31 cpu=1.37ms elapsed=5.24s tid=0x00007ff4f908a000
nid=0x9b0b waiting on condition  [0x000070000277e000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787a0d388> (a java.util.concurrent.ThreadPoolExecutor$Worker)
```

```
"pool-3-thread-2" #86 prio=5 os_prio=31 cpu=1.15ms elapsed=5.24s tid=0x00007ff4f908a600
nid=0x990b waiting on condition  [0x0000700002881000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787a0d978> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-3-thread-3" #88 prio=5 os_prio=31 cpu=1.20ms elapsed=5.24s tid=0x00007ff4f9830200
nid=0x660b waiting on condition  [0x0000700002984000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)
```

```
   Locked ownable synchronizers:
        - <0x0000000787a0df18> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-3-thread-4" #90 prio=5 os_prio=31 cpu=1.24ms elapsed=5.24s tid=0x00007ff4f908d000
nid=0x980b waiting on condition  [0x0000700002a87000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787a0e4b8> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"VM Thread" os_prio=31 cpu=8.43ms elapsed=35.57s tid=0x00007ff4f8d0eee0 nid=0x3f03
runnable

"GC Thread#0" os_prio=31 cpu=0.32ms elapsed=35.59s tid=0x00007ff4f8d02970 nid=0x3403
runnable

"G1 Main Marker" os_prio=31 cpu=0.06ms elapsed=35.59s tid=0x00007ff4f8d037c0 nid=0x3503
runnable

"G1 Conc#0" os_prio=31 cpu=0.04ms elapsed=35.59s tid=0x00007ff4f8d04520 nid=0x3703
runnable
```

```
"G1 Refine#0" os_prio=31 cpu=0.08ms elapsed=35.59s tid=0x00007ff4f8c27250 nid=0x4803
runnable

"G1 Service" os_prio=31 cpu=3.67ms elapsed=35.59s tid=0x00007ff4f8f01190 nid=0x4603
runnable

"VM Periodic Task Thread" os_prio=31 cpu=11.59ms elapsed=35.34s tid=0x00007ff4f8c58af0
nid=0x6403 waiting on condition

JNI global refs: 14, weak refs: 0
```

3.3.3 Discussion on simulation results

3.3.3.1 Discussion on implementation strategy

In this implementation, the Executor Service approach is chosen to realize the multithreaded running capability. The reason for using this method is due to it offers convenient API to create fixed number thread pool. Based on the understanding of the provided online document, Executor Service can easily control and run multithreads. In this programming task, all the threads are required to stop properly after 15 seconds simulation, therefore Executor Service approach is an ideal solution. Beyond this, the Task class in this implementation extends JDK's Thread class in order to make sure the created Task objects can be operated by Executor Service in multithreaded way. When the program starts to create Task objects, the run() method which is inherited from Thread class will be override. Thread.sleep() is used to simulate the task is run by CPU. As long as the Task object's sleep time is smaller than the assigned burst time then the while loop will continue until the total sleep time greater or equal to the burst time, then the end time will be set. Meanwhile, the system start time and end time will be monitored by a while loop. As long as the system end time subtracts the start time less than 15 seconds, the simulation will continue to process the tasks, otherwise Executor Service will shut down the simulation.

3.3.3.2 Discussion on simulation findings

According to the simulation result, not all tasks can be completed during the 15 seconds simulation. During the simulation #0, there were seven tasks completed, four tasks were interrupted, and 19 tasks were on the waiting list; in the simulation #1, there were eight tasks completed, four tasks were interrupted after 15 seconds, and 18 tasks were on the waiting list; in the simulation #3, eight tasks

were finished during the simulation, four tasks were interrupted after 15 seconds, and 18 tasks were on the waiting list.

3.3.3.3 Discussion on the output of *jstack*

The first line displays the timestamp and the second line informs about the JVM (see the following screenshots) [3]:

```
(base) YMacBook:~ yuyaoduan$ jstack -l 7794
2021-12-26 20:02:57
Full thread dump OpenJDK 64-Bit Server VM (17.0.1+12-39 mixed mode, sharing):
```

After this, we can find the Safe Memory Reclamation (SMR) and non-JVM internal threads [3]:

```
Threads class SMR info:
_java_thread_list=0x00007ff4f8f0b5a0, length=17, elements={
0x00007ff4f9800000, 0x00007ff4f982ce00, 0x00007ff4f9000600, 0x00007ff4fa83fa00,
0x00007ff4fa006800, 0x00007ff4fa006e00, 0x00007ff4fa008400, 0x00007ff4f9001200,
0x00007ff4fa840000, 0x00007ff4f9046a00, 0x00007ff4f9045c00, 0x00007ff4f904a800,
0x00007ff4f9805e00, 0x00007ff4f908f600, 0x00007ff4fa008a00, 0x00007ff4fa00c200,
0x00007ff4f9806400
}
```

After this, there are 13 JVM environment threads [3]:

```
"main" #1 prio=5 os_prio=31 cpu=9554.51ms elapsed=10.32s tid=0x00007ff4f9800000
nid=0x2103 runnable  [0x0000700001444000]
   java.lang.Thread.State: RUNNABLE
        at MultithreadedService.runNewSimulation(MultithreadedService.java:136)
        at MultithreadedService.main(MultithreadedService.java:228)

   Locked ownable synchronizers:
        - None

"Reference Handler" #2 daemon prio=10 os_prio=31 cpu=0.19ms elapsed=10.30s
tid=0x00007ff4f982ce00 nid=0x4203 waiting on condition  [0x0000700001b5a000]
   java.lang.Thread.State: RUNNABLE
        at java.lang.ref.Reference.waitForReferencePendingList(java.base@17.0.1/Native
Method)
        at
java.lang.ref.Reference.processPendingReferences(java.base@17.0.1/Reference.java:253)
        at java.lang.ref.Reference$ReferenceHandler.run(java.base@17.0.1/Reference.java:215)

   Locked ownable synchronizers:
```

```
        - None

"Finalizer" #3 daemon prio=8 os_prio=31 cpu=0.32ms elapsed=10.30s tid=0x00007ff4f9000600
nid=0x5503 in Object.wait()  [0x0000700001c5d000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(java.base@17.0.1/Native Method)
        - waiting on <0x0000000787f02f30> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:155)
        - locked <0x0000000787f02f30> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:176)
        at java.lang.ref.Finalizer$FinalizerThread.run(java.base@17.0.1/Finalizer.java:172)

   Locked ownable synchronizers:
        - None

"Signal Dispatcher" #4 daemon prio=9 os_prio=31 cpu=0.49ms elapsed=10.28s
tid=0x00007ff4fa83fa00 nid=0xa503 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Service Thread" #5 daemon prio=9 os_prio=31 cpu=0.13ms elapsed=10.28s
tid=0x00007ff4fa006800 nid=0xa303 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Monitor Deflation Thread" #6 daemon prio=9 os_prio=31 cpu=0.21ms elapsed=10.28s
tid=0x00007ff4fa006e00 nid=0xa203 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
```

```
        - None

"C2 CompilerThread0" #7 daemon prio=9 os_prio=31 cpu=94.46ms elapsed=10.28s
tid=0x00007ff4fa008400 nid=0x5c03 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"C1 CompilerThread0" #9 daemon prio=9 os_prio=31 cpu=94.11ms elapsed=10.28s
tid=0x00007ff4f9001200 nid=0x5e03 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"Sweeper thread" #10 daemon prio=9 os_prio=31 cpu=0.09ms elapsed=10.27s
tid=0x00007ff4fa840000 nid=0x5f03 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Common-Cleaner" #11 daemon prio=8 os_prio=31 cpu=0.20ms elapsed=10.23s
tid=0x00007ff4f9046a00 nid=0xa003 in Object.wait()  [0x0000700002372000]
   java.lang.Thread.State: TIMED_WAITING (on object monitor)
        at java.lang.Object.wait(java.base@17.0.1/Native Method)
        - waiting on <0x000000787f424f8> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@17.0.1/ReferenceQueue.java:155)
        - locked <0x000000787f424f8> (a java.lang.ref.ReferenceQueue$Lock)
        at jdk.internal.ref.CleanerImpl.run(java.base@17.0.1/CleanerImpl.java:140)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)
```

```
        at jdk.internal.misc.InnocuousThread.run(java.base@17.0.1/InnocuousThread.java:162)

   Locked ownable synchronizers:
        - None

"Monitor Ctrl-Break" #12 daemon prio=5 os_prio=31 cpu=65.20ms elapsed=10.07s
tid=0x00007ff4f9045c00 nid=0x9f03 runnable  [0x0000700002475000]
   java.lang.Thread.State: RUNNABLE
        at sun.nio.ch.SocketDispatcher.read0(java.base@17.0.1/Native Method)
        at sun.nio.ch.SocketDispatcher.read(java.base@17.0.1/SocketDispatcher.java:47)
        at sun.nio.ch.NioSocketImpl.tryRead(java.base@17.0.1/NioSocketImpl.java:261)
        at sun.nio.ch.NioSocketImpl.implRead(java.base@17.0.1/NioSocketImpl.java:312)
        at sun.nio.ch.NioSocketImpl.read(java.base@17.0.1/NioSocketImpl.java:350)
        at sun.nio.ch.NioSocketImpl$1.read(java.base@17.0.1/NioSocketImpl.java:803)
        at java.net.Socket$SocketInputStream.read(java.base@17.0.1/Socket.java:966)
        at sun.nio.cs.StreamDecoder.readBytes(java.base@17.0.1/StreamDecoder.java:270)
        at sun.nio.cs.StreamDecoder.implRead(java.base@17.0.1/StreamDecoder.java:313)
        at sun.nio.cs.StreamDecoder.read(java.base@17.0.1/StreamDecoder.java:188)
        - locked <0x0000000787e9e100> (a java.io.InputStreamReader)
        at java.io.InputStreamReader.read(java.base@17.0.1/InputStreamReader.java:177)
        at java.io.BufferedReader.fill(java.base@17.0.1/BufferedReader.java:162)
        at java.io.BufferedReader.readLine(java.base@17.0.1/BufferedReader.java:329)
        - locked <0x0000000787e9e100> (a java.io.InputStreamReader)
        at java.io.BufferedReader.readLine(java.base@17.0.1/BufferedReader.java:396)
        at com.intellij.rt.execution.application.AppMainV2$1.run(AppMainV2.java:49)

   Locked ownable synchronizers:
        - <0x0000000787e93d20> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)

"Notification Thread" #13 daemon prio=9 os_prio=31 cpu=0.07ms elapsed=10.07s
tid=0x00007ff4f904a800 nid=0x9e03 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
```

```
    - None
```

Each of them has same structure including the following part [3]:

1. Name: it refers to the thread name
2. Priority(prio): the priority of the thread
3. Java ID(tid): the unique id given by the JVM
4. Native ID(nid): the unique ID given by the operating system which can be used for getting correlation with CPU or memory processing
5. State: the actual state of the thread
6. Stack trace: following the state of the thread, it provides valuable source information to identify what is happening with the program

After this, there are four threads that are created by Executor Service, which are used for this simulation task. "pool-1" is related to the program's "numSimulations" attribute that refers to the first round of the simulation [3]. We can find out that all the JVM methods executed by this program behind the scenes [3]. Because of this, the developers are able to identify the root of the problem by looking at the source code or other internal JVM processing [3].

```
"pool-1-thread-1" #15 prio=5 os_prio=31 cpu=3.33ms elapsed=10.01s tid=0x00007ff4f9805e00
nid=0x9c03 waiting on condition  [0x000070000277e000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787edb900> (a java.util.concurrent.ThreadPoolExecutor$Worker)
```

```
"pool-1-thread-2" #17 prio=5 os_prio=31 cpu=3.35ms elapsed=10.01s tid=0x00007ff4f908f600
nid=0x9b03 waiting on condition  [0x0000700002881000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787edc208> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-1-thread-3" #19 prio=5 os_prio=31 cpu=2.48ms elapsed=10.01s tid=0x00007ff4fa008a00
nid=0x6603 waiting on condition  [0x0000700002984000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
```

```
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
        - <0x0000000787edc7a0> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"pool-1-thread-4" #21 prio=5 os_prio=31 cpu=2.46ms elapsed=10.01s tid=0x00007ff4fa00c200
nid=0x9803 waiting on condition  [0x0000700002a87000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(java.base@17.0.1/Native Method)
        at MultithreadedService$1.run(MultithreadedService.java:116)
        at
java.util.concurrent.Executors$RunnableAdapter.call(java.base@17.0.1/Executors.java:539)
        at java.util.concurrent.FutureTask.run(java.base@17.0.1/FutureTask.java:264)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.1/ThreadPoolExecutor.java
:1136)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.1/ThreadPoolExecutor.jav
a:635)
        at java.lang.Thread.run(java.base@17.0.1/Thread.java:833)

   Locked ownable synchronizers:
                - <0x0000000787edcd68> (a java.util.concurrent.ThreadPoolExecutor$Worker)
```

After this, at the end of the dump, the user could find several additional threads performing background operations such as Garbage Collection (GC) or object termination as following [3]:

```
"VM Thread" os_prio=31 cpu=3.36ms elapsed=10.30s tid=0x00007ff4f8d0eee0 nid=0x3f03
runnable

"GC Thread#0" os_prio=31 cpu=0.21ms elapsed=10.32s tid=0x00007ff4f8d02970 nid=0x3403
runnable

"G1 Main Marker" os_prio=31 cpu=0.06ms elapsed=10.32s tid=0x00007ff4f8d037c0 nid=0x3503
runnable
```

```
"G1 Conc#0" os_prio=31 cpu=0.04ms elapsed=10.32s tid=0x00007ff4f8d04520 nid=0x3703
runnable

"G1 Refine#0" os_prio=31 cpu=0.08ms elapsed=10.32s tid=0x00007ff4f8c27250 nid=0x4803
runnable

"G1 Service" os_prio=31 cpu=1.23ms elapsed=10.32s tid=0x00007ff4f8f01190 nid=0x4603
runnable

"VM Periodic Task Thread" os_prio=31 cpu=3.47ms elapsed=10.07s tid=0x00007ff4f8c58af0

nid=0x6403 waiting on condition
```

In the end, the dump will display the the Java Native Interface (JNI) references from which we can find out if memory leak exists or not since it could not be collected by GC [3], see the following table:

```
JNI global refs: 14, weak refs: 0
```

4. Reflection

This assignment provides a valuable chance to systematically study the filesystems of the operating system, typically the Linux B-Tree Filesystem (BTRFS). By reviewing all existed filesystems, the researchers point out the uniqueness, pros, and cons of BTRFS compared to others related filesystems. The research uses four main workloads to simulate servers' operations in real life that can be considered as a comprehensive model [2]. According to the results, BTRFS shows a good potential by using SSD [2], however, its disadvantages are quite obvious when working with HDD in terms of mail and web services. This experiment suggests that BTRFS is still a young system which has many aspects can be improved in the near future [2]. By utilizing BTRFS, the operating system can realize copy-on-write, efficient snapshots and strong data integrity that can be implemented from enterprise servers to smartphones and embedded systems which shows a board application prospect [2].

The implementation of multi-threaded service application is challenge part of this assignment that aims to build up the capability of understanding official documents, and then using the understanding to implement the program. By extending Java thread class and using executor service API, this implementation successfully achieved multi-threaded running. The thread sleep operation is used to simulate CPU processing tasks in actual situations. The research results shed lights on the

importance of multi-threaded programming in the current business activities since how to make a better use of multi-core CPU is the key to improving actual work's efficiency.

References

[1] Silberschatz A, Galvin P, Gagne G, Operating Systems Concepts (Tenth Edition), International Student Version, John Wiley & Sons, ISBN 978-1-118-06333-0.

[2] Rodeh, Ohad & Bacik, Josef & Mason, Chris. (2013). BTRFS: The linux B-tree filesystem. ACM Transactions on Storage (TOS). 9. 10.1145/2501620.2501623.

[3] Baeldung, "*How to Analyze Java Thread Dumps*", [2021-12-09], url: [https://www.baeldung.com/java-analyze-thread-dumps]