Java Update

2025 TPF Users Group Conference
May 4-7, Austin, TX

Applications Subcommittee

—

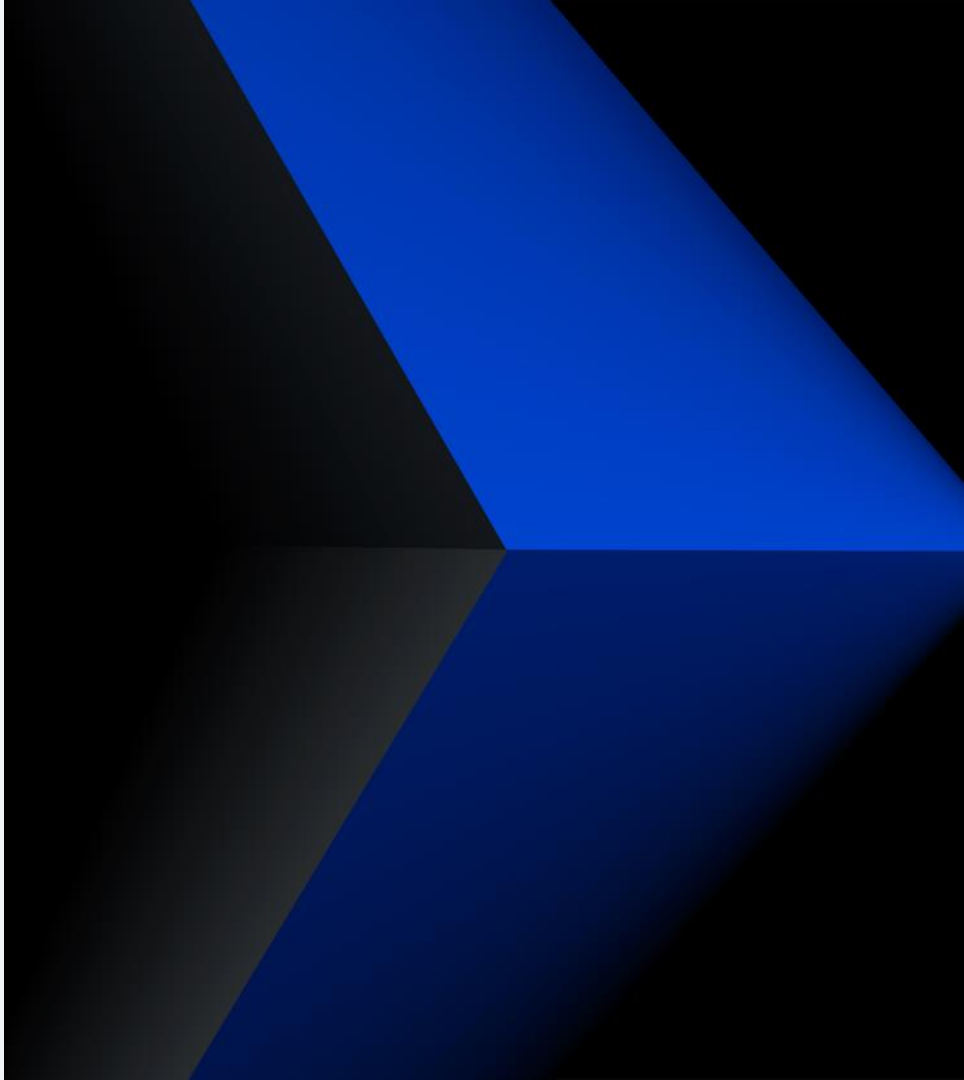
Daniel Gritter

IBM Z

IBM

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.



Semeru 21 (PJ48101 – April 2025)

z/TPF Release Calendar

Features

Migration

Semeru Runtime on z/TPF Release Calendar

LTS Java Version	Refresh Version *	Release Date	End Of Service
11	11.0.26.0 11.0.28.0 11.0.30.0 (last 11 release) Separate Drop 11 APAR	1Q2025 (PJ48152) 3Q2025 1Q2026 2H2026	2H 2026
21	21.0.6.0 (z/TPF GA for 21) 21.0.8.0 …	1H2025 (PJ48101) 3Q2025 …	TBD
25	TBD (1H 2026)	TBD	TBD

* Vulnerability deadlines permitting, the lab may skip a Refresh to lower maintenance burden for community

RandomAccessFile Class Caching

What is it?

The RandomAccessFile Class was updated to take advantage of existing JVM caching for Read-Only files. RandomAccessFile is used indirectly by other classes, including the JarFile class.

What was the issue?

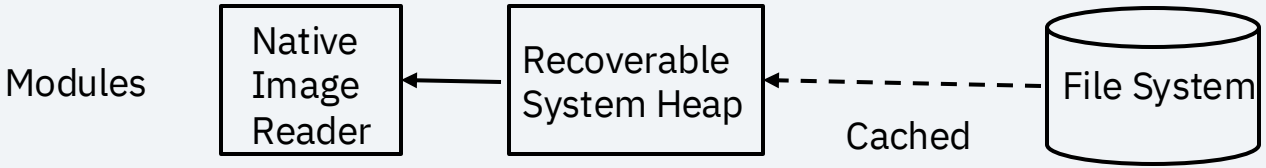
On z/TPF to read to an arbitrary position in a file, the file had to be read from the beginning of the file every time. For zip/jar files, the directory is at the end of the file (so that files can be added without rebuilding the entire file). Thus every time a class is read in the entire file would have to be traversed multiple times.

What is the benefit?

Startup/Recovery performance. Internal IBM testing observed an additional 20-25% improvement in JAM startup time on a z16.

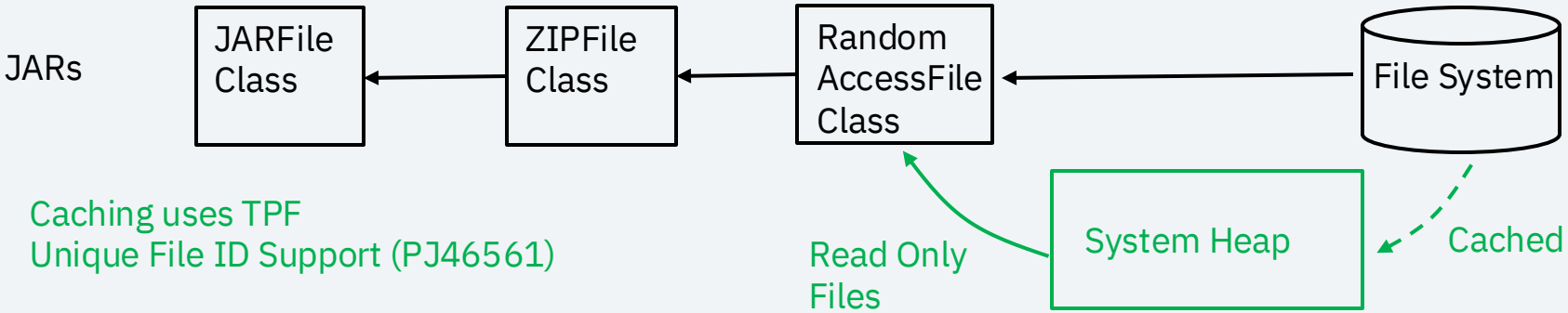
RandomAccessFile Class Caching

With Java 11 caching was introduced for some files



Before Semeru 21 (PJ48101)
After Semeru 21

With Java 21 caching was extended for read only JAR files (these contain Java applications / libraries)



Virtual Threads (new for Java 21)

What are they?

Java threads that can suspend running a Java task on a given OS thread to allow another Java task to run on the same OS thread. The suspend occurs at the Java layer during specific File System or Network operations.

What is the potential benefit?

Allows for certain workloads to create Java tasks to execute concurrently beyond the z/TPF threads per process limit (MTHD), requiring fewer ECBs allocated for Java processes.

Virtual Threads - How to create virtual threads

Virtual Thread Basic Example

```
public class Mytask implements Runnable {

  public static void main(String[] args) {
    Mytask obj = new Mytask();
    Thread virtualThread = Thread.ofVirtual.start(obj);
    System.out.println("Running outside of the thread");
    virtualThread.join();
  }

  public void run() {
    System.out.println("Inside virtual thread, running task.");
  }
}
```

Use unstarted() method to start thread later

Join will wait for thread to complete

Virtual Thread Executor Example

```
public class Mytask implements Runnable {

  ExecutorService taskManager = Executors.newVirtualThreadPerTaskExecutor();

  public static void main(String[] args) {
    Mytask obj = new Mytask();
    Future stillRunning = taskManager.submit(obj);

    stillRunning.get();  //Returns NULL if task completes
  }

  public void run() {
    System.out.println("Inside virtual thread, running task.");
  }
}
```

Implement Callable instead of Runnable to Work with responses from threads.

Minimal code updates required to convert existing Thread Executor to Virtual Thread Executor.

Use submit() method to get a future to check status without blocking.

Virtual Threads

Configuration Controls

jdk.virtualThreadScheduler.parallelism – optimal z/TPF threads the Virtual Thread Support will use, defaults to active I-stream count when the JVM starts.

jdk.virtualThreadScheduler.maxPoolSize – maximum number of z/TPF ECBs the process can use for virtual threads (default maximum is MTHD setting on z/TPF)

Virtual Threads - Diagnostics

For example viewing a call stack from javacore.20250115.082814.1179320404.0001.txt

```
3XMTHREADINFO3      Java callstack:
4XESTACKTRACE       at com/ibm/tpf/MQInputThread.getRequest(Native Method)
4XESTACKTRACE       at com/ibm/tpf/MQInputThread.run(MQInputThread.java:668(Compiled Code))
4XESTACKTRACE       at java/util/concurrent/Executors$RunnableAdapter.call(Executors.java:572(Compiled Code))
4XESTACKTRACE       at java/util/concurrent/FutureTask.run(FutureTask.java:317(Compiled Code))
4XESTACKTRACE       at java/lang/Thread.runWith(Thread.java:1608(Compiled Code))
4XESTACKTRACE       at java/lang/VirtualThread.run(VirtualThread.java:335(Compiled Code))
4XESTACKTRACE       at java/lang/VirtualThread$VThreadContinuation$1.run(VirtualThread.java:215(Compiled Code))
4XESTACKTRACE       at jdk/internal/vm/Continuation.enter(Continuation.java:186(Compiled Code))
3XMTHREADINFO3      No native callstack available on this platform
3XMTHREADINFO4          Type: Virtual, java/lang/CarrierThread:0x0000000523415510
```

// To view Virtual Thread Diagnostics in Java Corefile two options required
-XX:+ShowCarrierFrames
-XX:+ShowUnmountedThreadStacks

Virtual Threads

Recommendation

Lab will watch for opportunities to apply virtual threads when it makes sense and will ensure a seamless transition as necessary.

We will investigate the current use of Thread pools and migrate to virtual threads where applicable.

Other platforms typically see most benefit from using virtual threads when running thousands of concurrent requests

Other New Features between Java 11 to Java 21

- **Text Blocks**
- **Records**
- Sealed Classes
- Instanceof method extensions
- NULL pointer exception extensions
- Context Specific Deserialization
- Day Period Support
- Stream.toList()

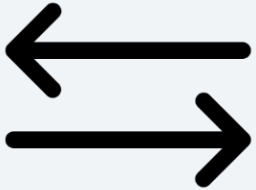
Good for working with immutable fixed field data structures.

```
String text = """
        {
           "name": "John",
           "age": 30,
           "city": "New York"
        }
        """;
```

Great for working with JSON Docs!

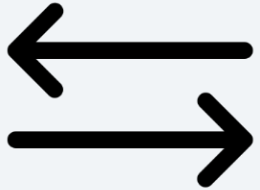
```
public record Person(String name, int age, String city) {
}
...
Person person = new Person("John", 30, "New York");
System.out.println("Name: " + person.name());
System.out.println("Age: " + person.age());
System.out.println("City: " + person.city());
```

Migration Considerations to Semeru 21



- Semeru 11 and Semeru 21 can coexist on z/TPF.
- Semeru 21 and IBM Java 8 cannot coexist on z/TPF.
- Semeru 11 compiled applications can run in Semeru 21 runtime
- jam.env file is used to control default version of the Java Runtime used across all JAMs.
- Each JAM descriptor can be overridden to specify a different version of Java Runtime to run in.
- zfile environment shell PATH setting only controls Version of the Java command line.

Migration Considerations to Semeru 21



- JDK Community deprecated the Security Manager
- The Security Manager as of version 21 is available, but compilation and runtime will get deprecation warnings.
- z/TPF is keeping the class loading security aspect independently of the Java Security Manager deprecation.
  - For Java 11 z/TPF default options specify a custom security manager using the java.security.manager property: “-Djava.security.manager=com.ibm.tpf.TPFSecurityManager”.
  - For Semeru 21, class loader security is on by default. To disable it, you may set the com.ibm.tpf.disableSecurityManager property (“-Dcom.ibm.tpf.disableSecurityManager”) either globally or on a per JAM basis. This is an existing property used for either version to disable the TPF classloading protection.

Secure FTP

PJ46830 (July 2023) introduced SFTP support leveraging a SSH server package (Apache MINA).

IBM is currently investigating a new SFTP client for z/TPF based on the same package.

We would like feedback on what authentication methods you use with SFTP clients and how you perform key management (if used) today. Interested in being a sponsor user? Contact Dan Gritter (dgritter@us.ibm.com)

SFTP enhancements

In addition to the SFTP client, IBM is investigating enhancements to the SFTP server:

- Operation Logging
- Offboard authentication (LDAP)
- ASCII mode transfers (requires updates to the MINA source code / contributions to open source)

Have other requirements? Contact Dan Gritter (dgritter@us.ibm.com) or open an IBM IDEA!

Thank you