Java on z/TPF – a retrospective and analysis

2023 TPF Users Group Conference
April 24-26, Dallas, Texas
Application Development

—

Daniel Gritter

Disclaimer

History of Java on z/TPF

Problem Statement

In March 2015, we formally announced to clients our intention to bring Java onto the z/TPF platform. We had 3 main goals:

- Attracting new talent / expand available developer pool
- Increase business agility
- Integrate new technology

From the beginning we used design thinking to get continuous client input.

Skills Availability

We had feedback from clients that it was hard to attract new developers to z/TPF. Java was an excellent opportunity to solve aspects of this challenge

- Abundant Java skills in the marketplace
- IBM’s commitment to enterprise Java
- Development tooling and offline development

Business Agility

Integrating Java components into the application stack can provide significant dividends in terms of development time and flexibility – some key technologies are

- Integration of business rules engine to replace application rules
- Enterprise integration (Kafka, MongoDB)
- Application orchestration (Reducing latency of outside orchestration)

Integrate new technology

Adding Java provided rapid development capabilities and integration libraries that the lab believes is invaluable in helping customer re-architect and expand their applications

- Micro-services approach
- Multiple programming models
- Wealth of standard enterprise-grade libraries for integration

Delivering Java

Using customer feedback and design approach – we came up with some basic criteria for a solution:

- Keep “TPFisms” out of Java
  - Sandbox to reduce exposure to potential Java vulnerabilities
- Defining with customer input primary focus of TPF calling Java vs Java calling TPF
- Must be certified and supported Java

Delivering Java

Getting Java deployed on z/TPF took a lot of effort and collaboration from the z/TPF team. The major efforts were

- Porting the Java virtual machine to z/TPF
- Paradigm shift for loading / deploying applications
- Long running Java process vs traditional ECB model
- Managing deployments to be scalable and reliable
- Interfacing Java with TPF applications in a way that was natural to both TPF and Java developers

Java GA on z/TPF

We announced support for Java on z/TPF at the 2017 TPF Users Group conference.

We were pleased to announce the title “IBM 64-bit Runtime Environment for z/TPF Java™ Technology Edition, Version 8”

This release included support for the Java runtime, local TPF as well as remote invocation of Java services, and the introduction of the JAM infrastructure, providing for redundant and scalable Java services.

Java GA on z/TPF

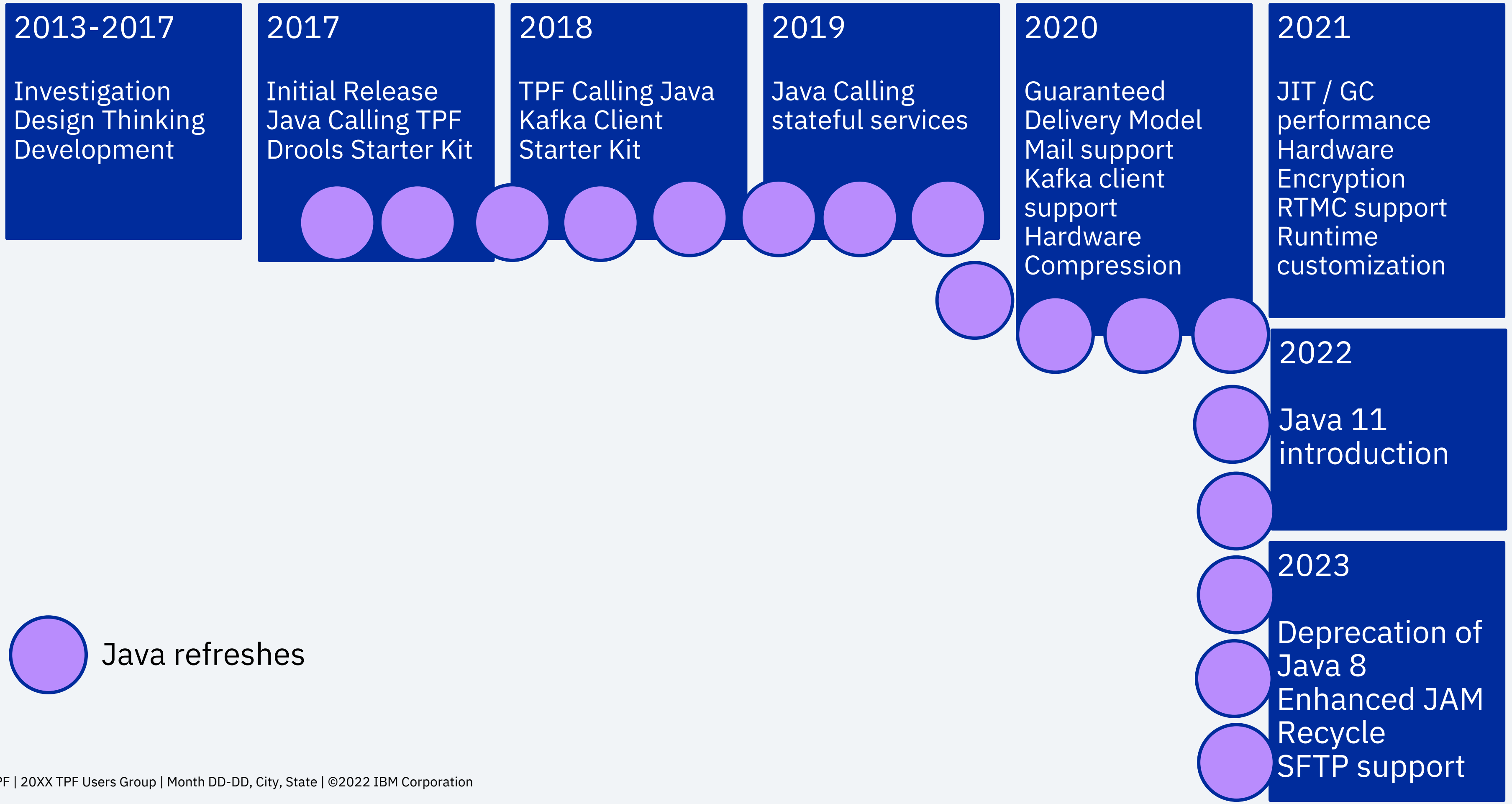
With this release we envisioned co-existence of Java and TPF applications and sibling services

- Defined interface between the different languages
- Load balancing and optimization handled by system
- Extensibility for future flexibility (introducing ability to move services on and off TPF to match business needs)

Follow-on efforts

With the delivery of Java complete we didn’t stop ther - we continued to extend support to deliver a lot of additional features based on customer need:

- ❖ Consistent delivery of quarterly refreshes
- ❖ Java calling TPF code
- ❖ Java monitoring and diagnostics
- ❖ Guaranteed Delivery support
- ❖ Hardware compression + encryption support
- ❖ Starter kits highlighting Java solutions
- ❖ Performance improvements



2013-2017
Investigation
Design Thinking
Development

2017
Initial Release
Java Calling TPF
Drools Starter Kit

2018
TPF Calling Java
Kafka Client
Starter Kit

2019
Java Calling
stateful services

2020
Guaranteed
Delivery Model
Mail support
Kafka client
support
Hardware
Compression

2021
JIT / GC
performance
Hardware
Encryption
RTMC support
Runtime
customization

2022
Java 11
introduction

2023
Deprecation of
Java 8
Enhanced JAM
Recycle
SFTP support

● Java refreshes

IBM delivering direct value with Java

Beyond just supporting customer applications, Java has proved critical in supporting several TPF functions and will continue to add new value to the platform:

- MongoDB client for remote subfile
- Mail daemon replacement
- Kafka Producer
- Native code coverage server
- SFTP server

System-wide Java monitoring

As the IBM lab continued to introduce features that depend on Java the RTMC tooling emerged as the new standard for collecting metrics on z/TPF

JVM Monitoring provides details at the JVM, JAM, and system-wide level integrated with native TPF statistics to provide a comprehensive analysis of system health and performance.

Results of Java on z/TPF

Java Proof points

Case Study: MongoDB client

This example is used from developing the Remote Subfile project. Compared with the cost of porting and testing the MongoDB client as a native library

Using Java:

Functional service to connect to Kafka in a matter of weeks

JAM model already exists for integration with native TPF components

Extra time available to optimize interface between z/TPF and Java

Easy upgrade path to new versions of the package – bug fix easily incorporated towards the end of development

Porting Native library:

Estimated multiple year project with multiple platform dependencies

Process model changes necessary to optimize for TPF

Additional testing effort necessary due to porting complexity

Using the Java package for connectivity to MongoDB servers resulted in quick delivery of function with a high confidence in overall quality

Java Proof points

Case Study: Operational Decision Management on z/TPF

This example is extending a z/TPF application with a call to a Rules engine. A comparison is made between calling ODM on z/TPF running within a JAM to ODM running on a collocated Linux on IBM Z LPAR. In this example the message rate as 500 requests / sec going to ODM. Results show the cost to run the rules engine with the sample rules to be less than the cost of making the call outbound, even without TLS.

ODM on z/TPF	ODM on Linux on IBM Z
Average Latency: 0.5 ms (0.19 ms min 23 ms max)	Average Latency: 3 ms (1.5 ms min 75 ms max)
TPF CPU consumption: 3 I/S running at 10% utilization (5% GP, 5% TE)	TPF CPU consumption: 3 I/S running at 30% utilization (20% GP, 10% TE)
Linux CPU consumption: 0%	Linux CPU consumption: 1 I/S running at 50% utilization

Running ODM on z/TPF provides 6x faster average response time at ~5x less total CPU utilization, and ~3x less total TPF CPU utilization

Java Proof points

Case Study: Kafka Connect vs TPF Guaranteed Delivery

This example is comparing offloading TPF data to a Kafka broker directly from z/TPF as opposed to using Kafka Connect to pull TPF data from an MQ queue..

Guaranteed Delivery solution	Kafka Connect on Linux on IBM Z
(cpu utilization to process 1000 1k msgs)	(cpu utilization to process 1000 1k msgs)
TPF CPU consumption: 4.6% CPU	TPF CPU consumption: 6.6% CPU
0.8% TE CPU	0.6% TE CPU
Linux CPU consumption: 0.55 CPU%	Linux CPU consumption: 3.1% CPU

Running GD connectivity to Kafka on z/TPF resulted in the ability to offload messages at an almost 20% reduction in TPF MIPS and an almost 6x reduction in Linux MIPS.

Support and Compatibility of Java packages

The question has come up several times about “support” for Java packages on z/TPF.

As an officially certified Java environment, code that is completely written Java is fully compatible with the z/TPF Java runtime. Code that depends on JNI (Java Native Interface) routines will likely not function correctly on z/TPF unless those JNI routines are ported to the platform.

This means we expect the behavior of Java applications to be the same on z/TPF as other platforms, and clients should open a z/TPF support case in the event of any observed compatability issues.

Supported Java packages

There’s three different type of support the lab will provide for Java code packages:

- For Java packages owned by IBM, a support case opened with the TPF Lab will be routed to the right area owners whom we will work with to identify the issue and solution.
- For Java packages owned by the open-source community that we have dependencies on, we will investigate any issues and work with the appropriate team to either pick up a new version of the package or develop a fix for the problem identified.
- For other Java packages, we will make a best effort to assist with any problems but can only ensure they have similar behavior running on z/TPF as they do on other platforms. Bugs in software packages not owned or depended on by IBM products will ultimately have to be resolved by the package owner.

Looking forward at Java on z/TPF

Java going forward

We will be dropping support for Java 8 with a focus of transitioning to OpenJDK for Java 11 and beyond. We expect this to be a painless transition for any customer applications, and the work needed for IBM applications is already completed.

We expect clients to be able to migrate to Java 11 with minimal effort, in most cases without having to make any code updates.

Remember to complete the transition to Java 11 (already available) by the **end of 2023**, at which point we will no longer ship quarterly refreshes for Java 8.

Java going forward

As part of the transition to OpenJDK we expect to be able to focus additional effort on improvements to Java, particularly in the following areas:

Recovery – reduce the system impact to restart Java after a system outage

Startup – reduce the overhead needed to start Java upon system activation

Integration – better integration with TPF functionality (for example no longer requiring recycle on activation of programs or files not relevant to a running JAM)

Monitoring – capture additional data to assist in problem determination or prediction

Call to Action

Install / Use RTMC for production workloads

Investigate / Resolve any outstanding Java 8 dependencies in customer applications

Try out some of the features that leverage Java runtime (Mail, Kafka, SFTP server) provided by IBM with the product

Request for feedback

What are things you like about Java support on z/TPF

What are things you find challenging or frustrating with Java on z/TPF

What are some additional use cases for Java that IBM can assist with?

We want sponsor users!

Our development cycle is driven by your feedback.

We are looking for sponsor users to assist in design and implementation, targeting the following personas:

- System Monitoring

We expect to begin engaging with the sponsor users in 2H2023

If you are interested in participating as a sponsor user, please contact:

Daniel Gritter (dgritter@us.ibm.com)

Thank you