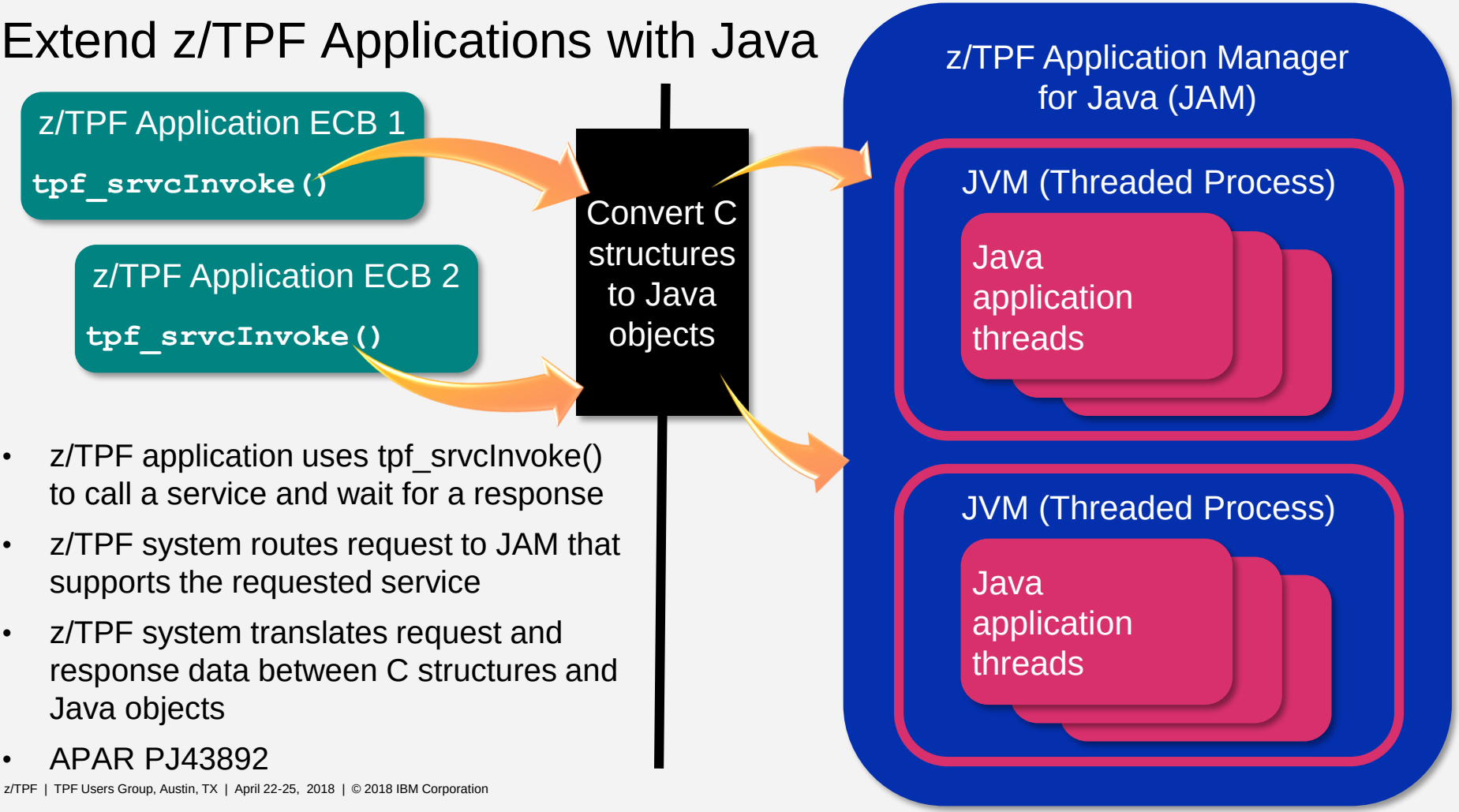
z/TPF Support for Java™ Enhancements

Chris Filachek
z/TPF and z/TPFDF Architecture & Development

What can I do with Java on z/TPF TODAY?

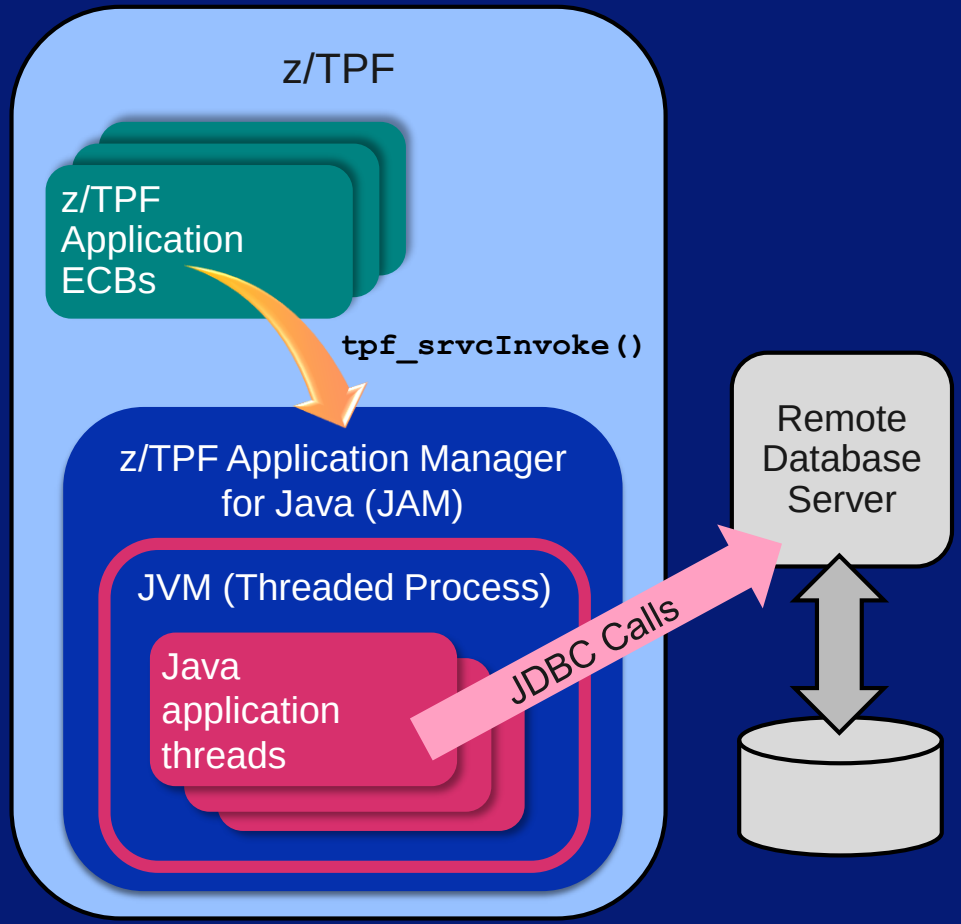
Extend z/TPF Applications with Java



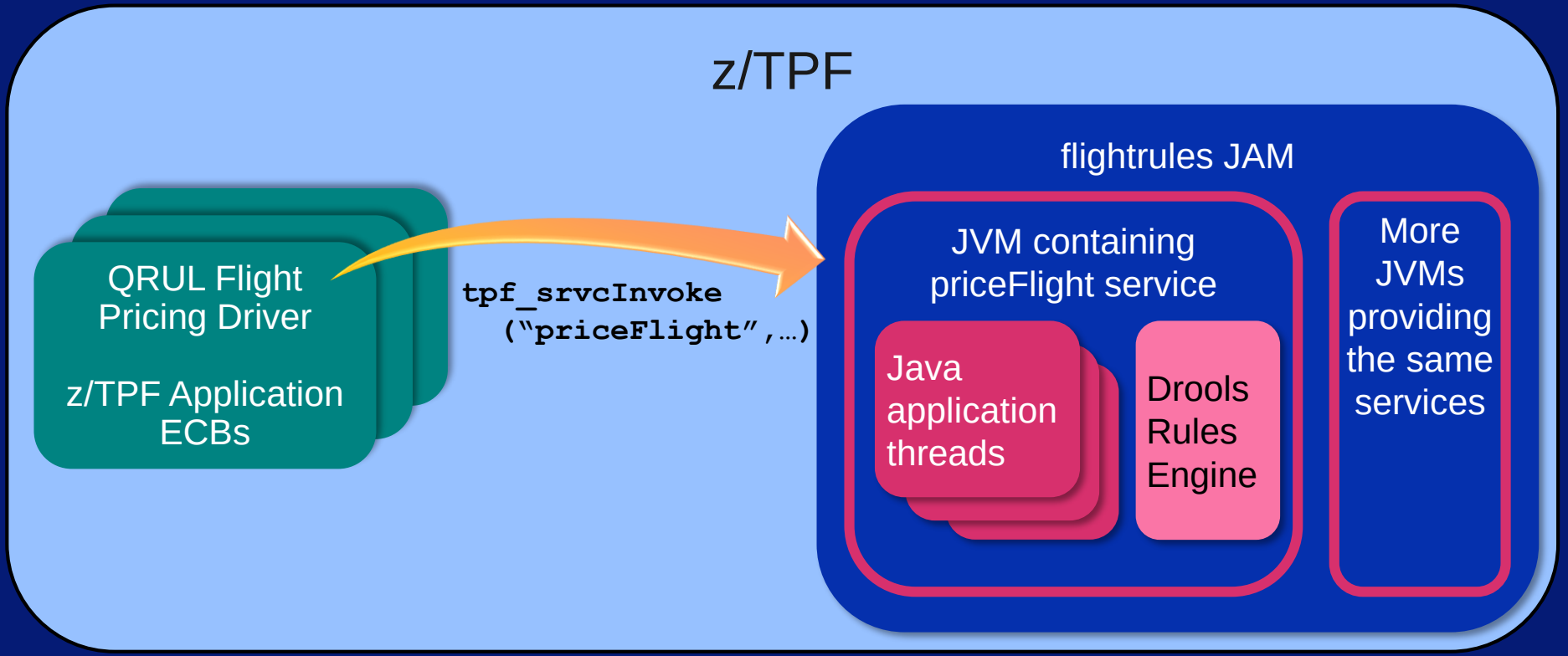
- z/TPF application uses tpf_srvcInvoke() to call a service and wait for a response
- z/TPF system routes request to JAM that supports the requested service
- z/TPF system translates request and response data between C structures and Java objects
- APAR PJ43892

How will you extend your apps with Java?

- Use JDBC or other database packages to import and export data to / from remote databases
  - Initialize or refresh z/TPF data without tapes or customized interfaces
  - Export z/TPF data from data events, utilities, or applications
- Simplify business logic and avoid network overhead by calling a rules engine on z/TPF
- Move existing enterprise business logic onto z/TPF and avoid network overhead
- Something else?

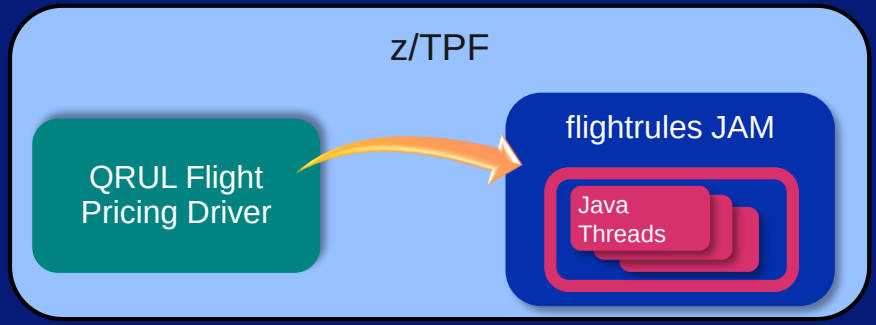


Our Example: z/TPF Rules Engine Driver for Java

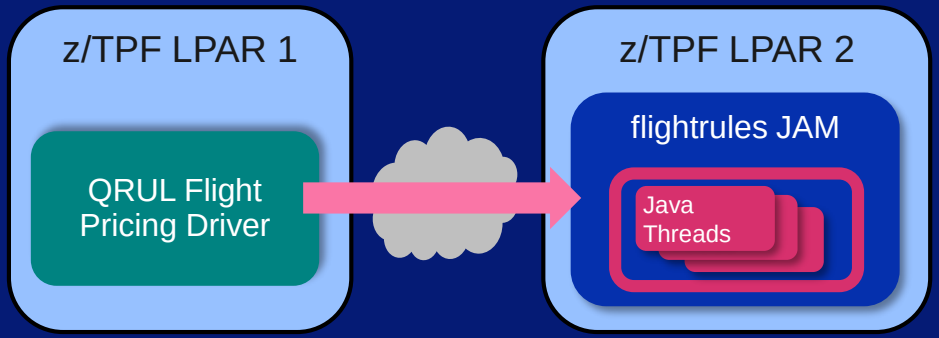


Performance Test: Local vs. Remote Rules Engines

- Local
  - z/TPF application and rules engine on the same z14 LPAR with 1 I-stream
  - 5043 requests / sec (actual)
  - ~99% of MIPS were Transformation Engine (TE) eligible

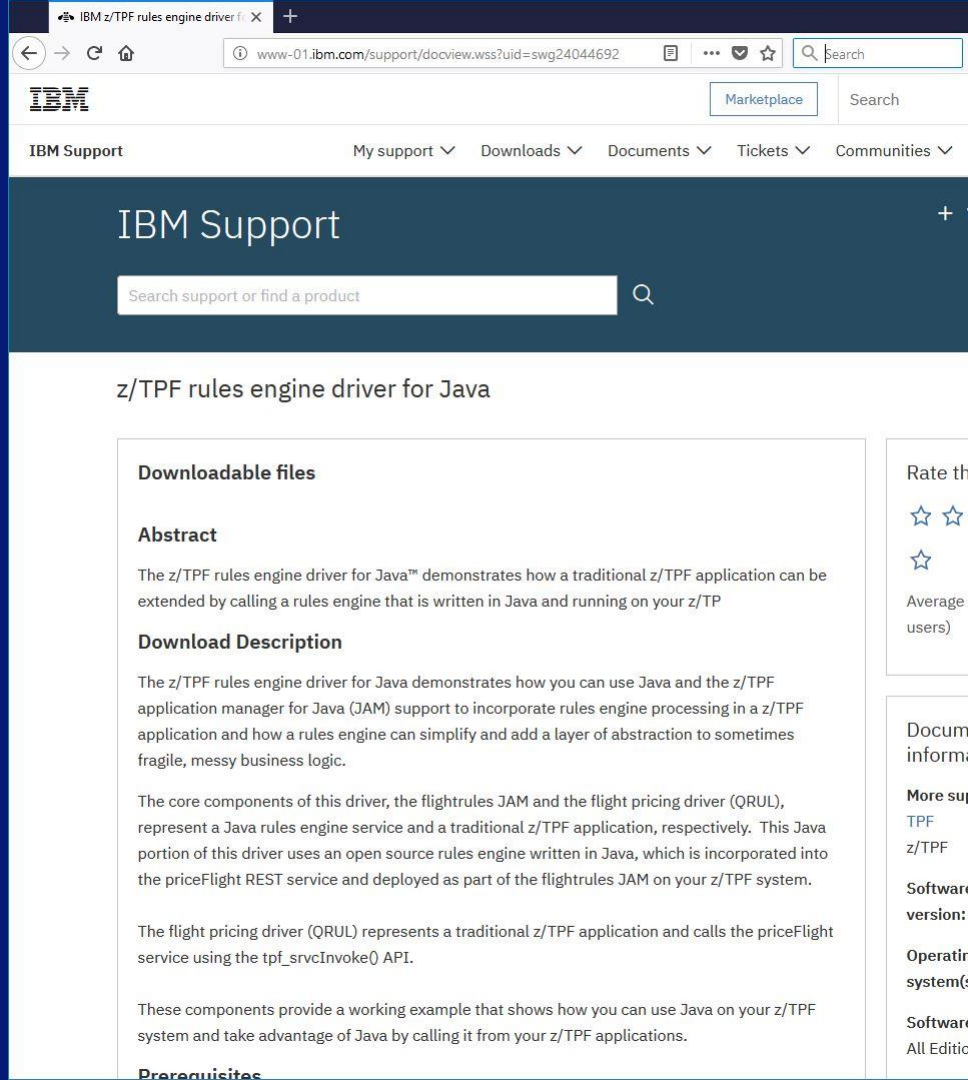


- Remote
  - z/TPF application and rules engine on separate z14 LPARs with persistent sockets
  - Response time is ~3x slower per request
    - At least 5x slower if rules engine is on physically separate hardware
  - ~81% of MIPS were TE eligible
  - 52% more MIPS to run the same workload
  - 78% more MIPS to run the same workload using encryption (SSL)

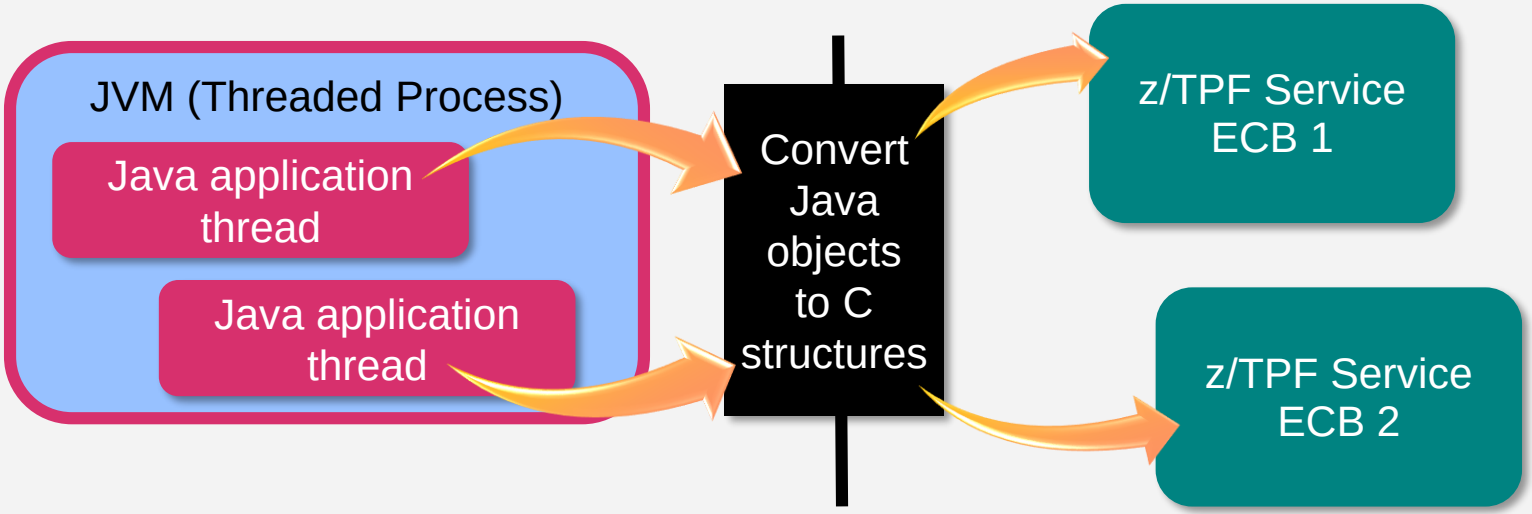


Interested in the z/TPF Rules Engine Driver for Java?

- Run our rules engine driver on your z/TPF test systems!
- See how to create a simple service in Java and call it from a sample application
- Available for download from the IBM z/TPF driver download site
  http://www-01.ibm.com/support/docview.wss?uid=swg24044692

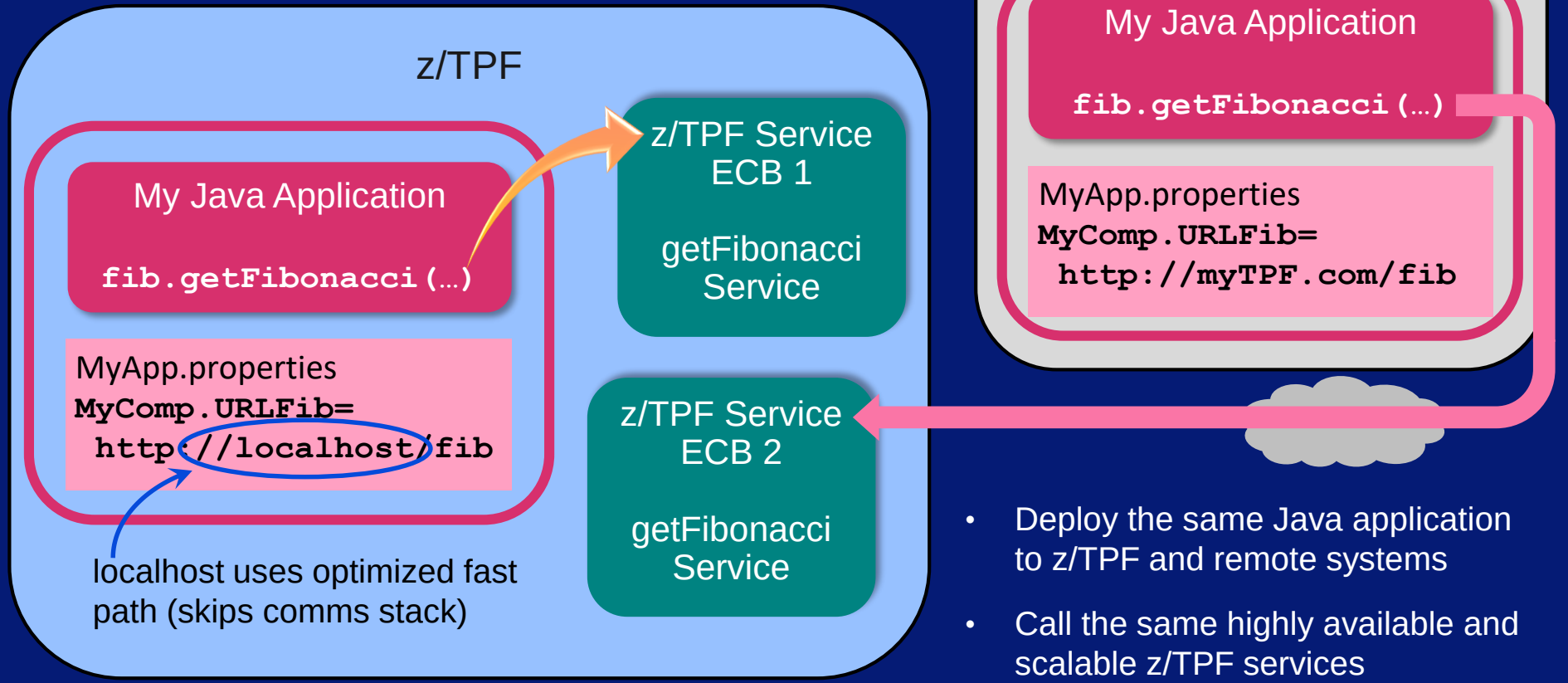


Call z/TPF Services from Java



- Java calls z/TPF services using an optimized, local REST calls
- z/TPF system creates a new z/TPF ECB to process the service request and respond to the Java caller
- z/TPF system translates request and response data between Java objects and C structures
- APAR PJ44844

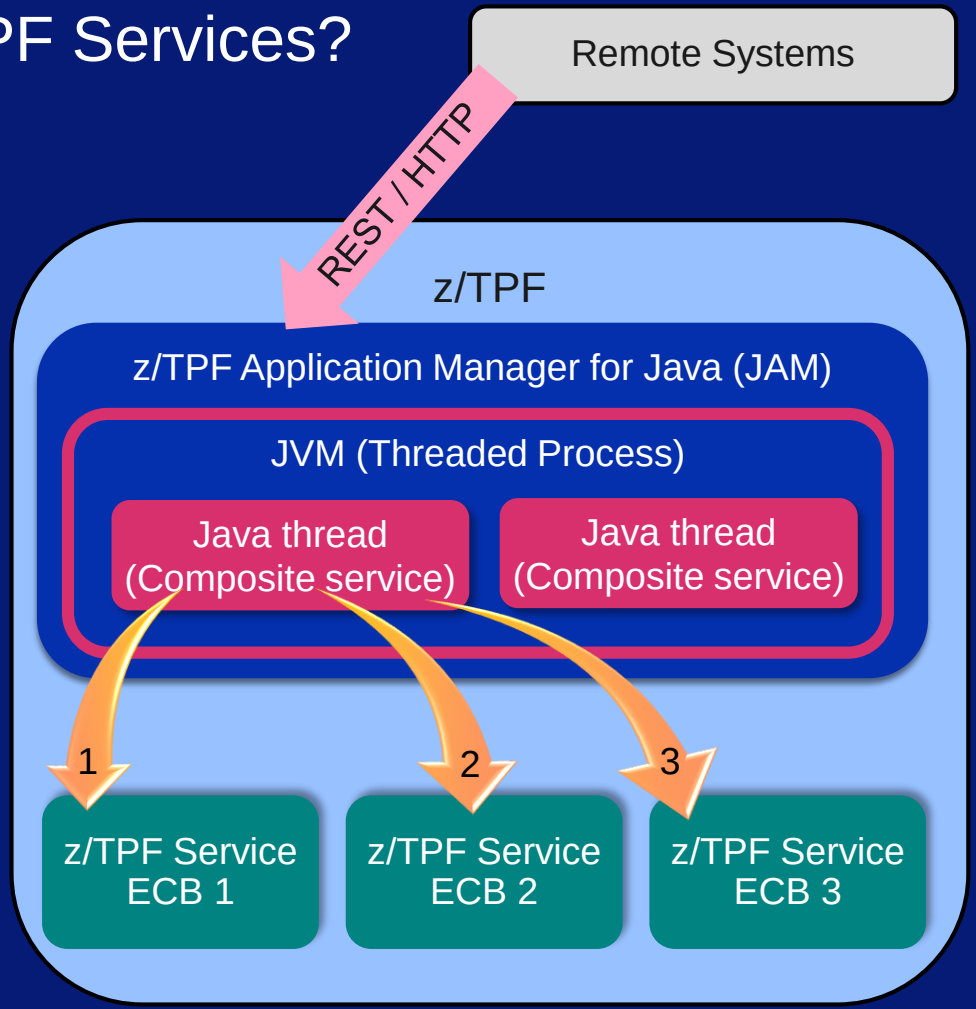
Deploy Java across your Enterprise and take advantage of z/TPF Services



- Deploy the same Java application to z/TPF and remote systems
- Call the same highly available and scalable z/TPF services

How will you use Java with z/TPF Services?

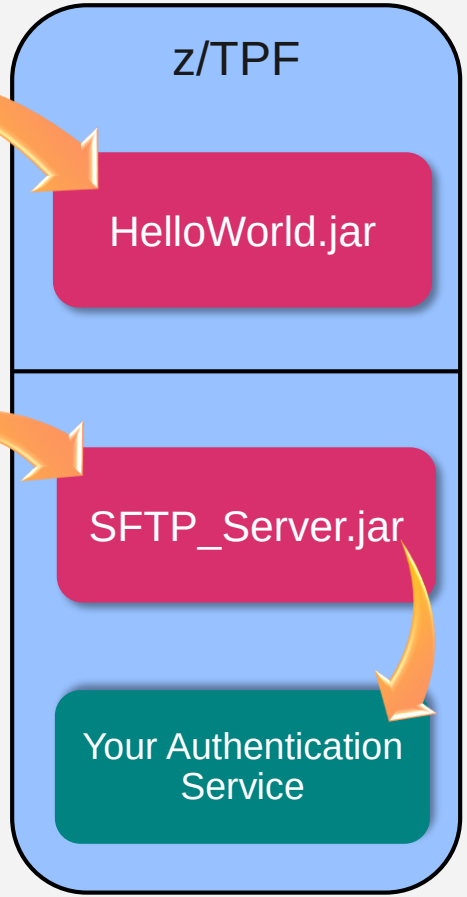
- Service orchestration?
  - Create a composite service using Java that calls local z/TPF microservices
  - Avoid network overhead of putting orchestration layer on other systems
- Extend Java applications with z/TPF services
  - Call your authentication services for Java applications
  - Access z/TPF data
- Something else?



Create standalone Java Applications and Utilities

```
AAES0008I 00 ==> zfile java -jar HelloWorld.jar
CSMP0097I 14.54.45 CPU-B SS-BSS  SSU-HPN  IS-01
FILE0001I 14.54.45 START OF DISPLAY FROM java -jar
/sys/tpf_pbfiles/bin/Hell...
Hello World from Java(TM)
END OF DISPLAY+

AAES0008I 00 ==> zfile java -jar SFTP_Server.jar
```



- Use ZFILE java command to run Java applications and utilities on z/TPF
- Extend applications and utilities with z/TPF services
  - For example: Run Java SFTP server on z/TPF and access your authentication services as a z/TPF service
- APAR PJ43892

What could I do with Java on z/TPF TOMORROW?

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.

What's next for Java on z/TPF?

- ✓ z/TPF applications calling Java
- ✓ Java calling z/TPF services
- ✓ Standalone Java applications and utilities
- ? Combine Java business logic and database updates
  - ? Simple: Interleave Java business logic with an update to a single database
  - ? Complex: Interleave Java business logic with updates to multiple databases

Simple Database Update

1. Start in Java
2. Read and Hold PNR
3. Java business logic
4. Update and File/Unhold PNR

Complex Database Update

1. Start in Java
2. Begin transaction scope
3. Open Bob's Account with Hold
4. Open Chris' Account with Hold
5. Java Business Logic (Transfer funds from Bob to Chris)
6. Update and Unhold Bob's Account
7. Java Business Logic (Pay college tuition for Chris' kids)
8. Update and Unhold Chris' Account
9. Commit transaction scope

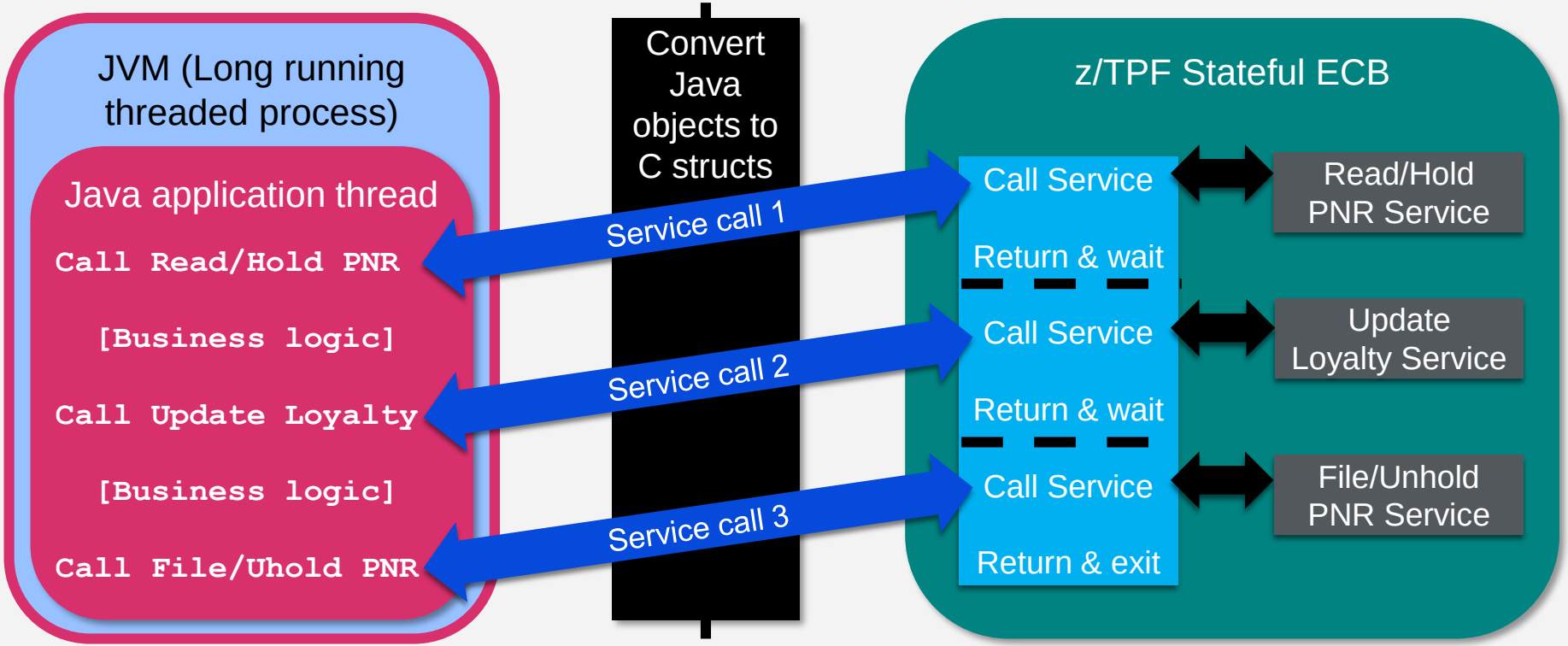
Java Hills

An application architect can use Java to incrementally modernize applications in place on z/TPF by leveraging existing high performing z/TPF databases and services.

A Java application programmer can use a Java object model to access and update z/TPF databases in a consistent manner without having to know z/TPF database constructs and can develop in half the time of legacy application programming models.

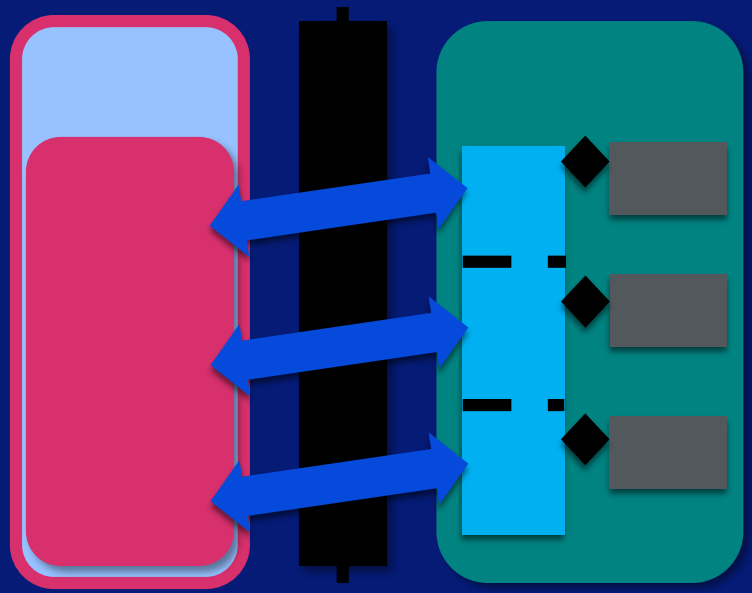
Java calling z/TPF Stateful Services

- Provide ability to make multiple calls from Java to the same z/TPF ECB.
- The z/TPF ECB can hold resources across service calls on behalf of Java
- Java can execute business logic between service calls.



Java calling z/TPF Stateful Services – Current Thinking

- Built on top of existing support for Java calling z/TPF services
  - Java calls a local REST service defined as stateful
  - Stateful services created with REST provider support
- Java programmers invoke stateful services using REST
  - REST service calls are a common programming model in Java
  - Java programmers do not have to know z/TPF database constructs or conventions
- Java + REST: Develop the application anywhere
  - Develop and using your favorite Java IDE
  - Test Java application by calling REST services deployed on z/TPF test system



Creating z/TPF Stateful Services – Current Thinking

- Stateful services created by you to meet the needs of your business
  - Services that read, hold, and update z/TPF Find/File and z/TPFDF databases
    - For example: ReadAccount service to read and hold a z/TPFDF account subfile
    - For example: ReadPNR service to read and hold both a Find/File PNR and the z/TPFDF PNR extension
  - Read or update memory areas (globals, system heap, etc.)
  - Use transaction scopes across service calls
- Services maintain database consistency
  - Access databases using the same methods as existing z/TPF applications

Creating z/TPF Stateful Services – Still More Thoughts…

- Stateful services are based on Java calling z/TPF services
  - z/TPF system converts request and response data between Java objects and C structures
  - No need to write data transformation logic
- Stateful services created by z/TPF programmers
  - z/TPF programmers manage z/TPF constructs and structures on behalf of the caller
  - z/TPF programmers do not have to know Java
- Separate Java and z/TPF ECBs provide easy cleanup of the z/TPF environment
  - Java processes are usually long running – a single Java thread can process many messages
  - Separate ECBs allow Java to start with new ECBs for each message

Calling all Sponsor Users!

Help us make sure our enhancements solve your needs!

Contact us if you are interested and not already participating in our sponsor users calls.

- Java calling z/TPF Stateful Services

Thank You!

Trademarks

Notes

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.