Guaranteed Delivery for JVM DF Queue support

—

Daniel Gritter
z/TPF Application Squad Lead

IBM Z

Disclaimer

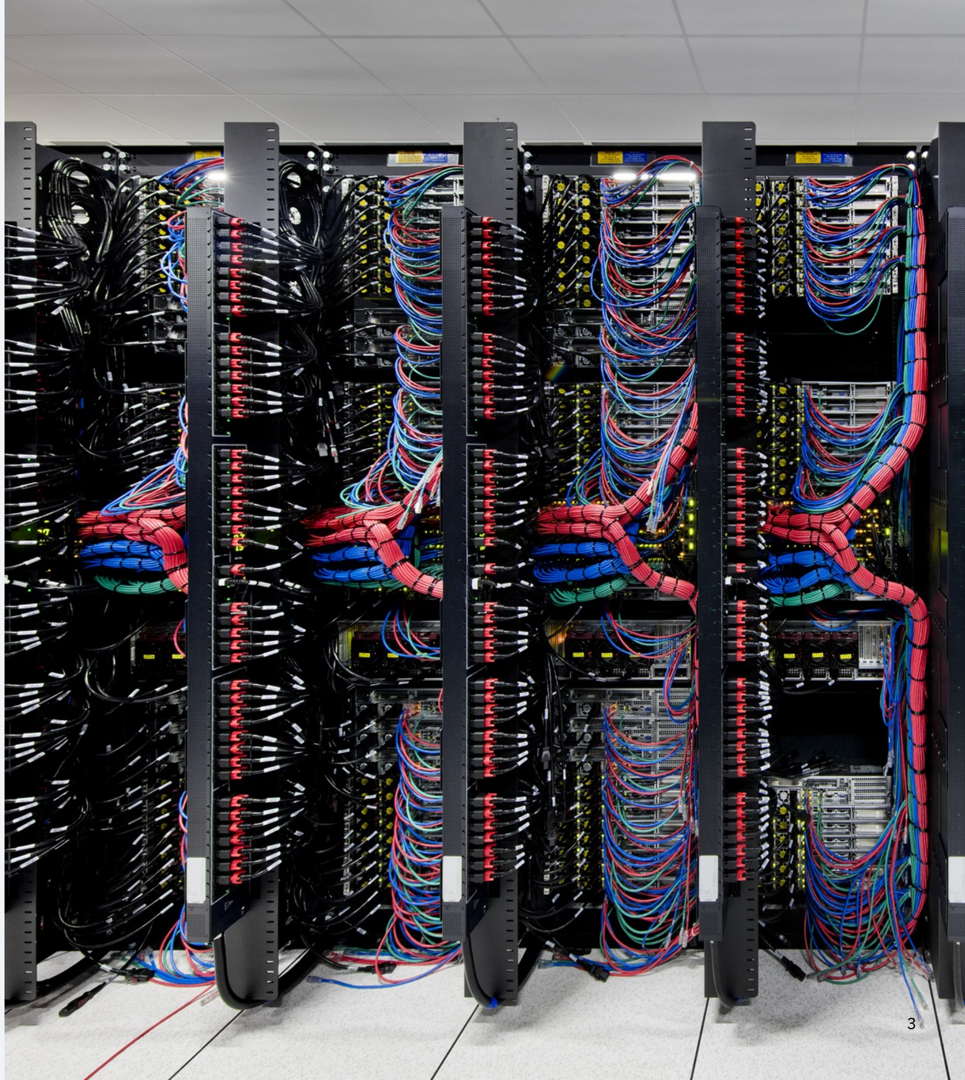
Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Background – Guaranteed Delivery for JVM

PJ45923 (March 2020) provided a new mechanism for publishing data using a Java application.

The initial deliverable provided a built-in Kafka producer support, with SMTP support following in APAR PJ46000 (May 2020).

This support required the use of MQ as the transport mechanism and retry / error processing.



Problem Statement

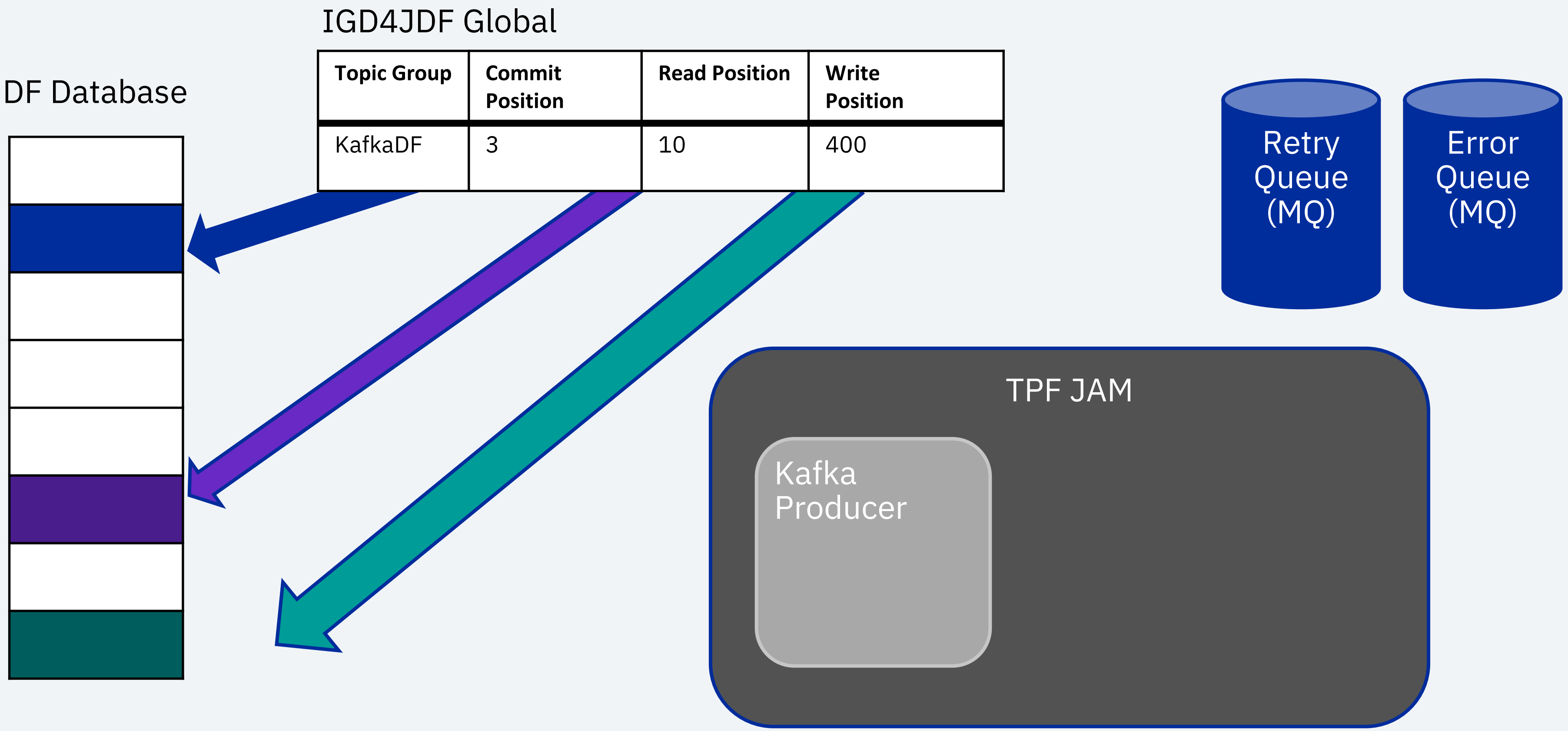
Resource and scalability limits inhibits utilization of MQ as a transport mechanism when sending a large number of small messages

Value Statement

Anna, the application architect, needs to log data through Kafka as part of a mainline z/TPF application

- Zach, the application programmer, can update his application to use Guaranteed Delivery service via a simple function call
- Calvin, the capacity planner, can account for the storage requirements of a high-volume throughput without using constrained resources
- Sophie, the system programmer, can configure the guaranteed delivery support to reach transmission rates of over 30,000 msgs/sec when publishing 4k messages

Technical Details



Topic Group	Commit Position	Read Position	Write Position
KafkaDF	3	10	400

TPF DF Database queue

Index only database that can reside in VFA* for fastest processing (processor unique using partition / interleave support)

*comes with the risk of data loss if VFA is not recovered over IPL

Choosing MQ vs DF queue support

MQ queues:
Standard tooling / queue management
Consistent behavior regardless of message size
Flexible queue insertion

DF queues:
Optimized for 4k and under message sizes
Must use tpf_publish_data api
Highest throughput when using VFA delay file

MQ vs DF Performance (2 I/S system, 3700 byte messages):

MQ Persistent
6,500 msgs/s at 30% utilization -> 13,000 msgs/s at 60% utilization

MQ Non-persistent
6,500 msgs/s at 30% utilization -> 17,000 msgs/s at 75% utilization

DF VFA Immediate
5,000 msgs/s at 50% utilization max throughput

DF VFA Delay file
20,000 msgs/s at 70% utilization scaling up to >30,000 msgs/s (hit OSA 1Gb throughput)

Recommendations for choosing MQ or DF queue

1) For persistent messages, use MQ

2) For non-persistent messages over 4k, use MQ

3) For non-persistent messages that can fit in a single 4k subfile, use DF

IGD4JDF global

Processor unique, keypointed global used to expedite recovery of position on IPL

Commit: database position indicating already delivered messages
Read: database position of next message to publish
Write: database position to place next message

Retry / Error queue

MQ continues to be used for error processing and retry processing.

New configuration option “RetryBlock” to define the behavior when there are items on the Retry queue.

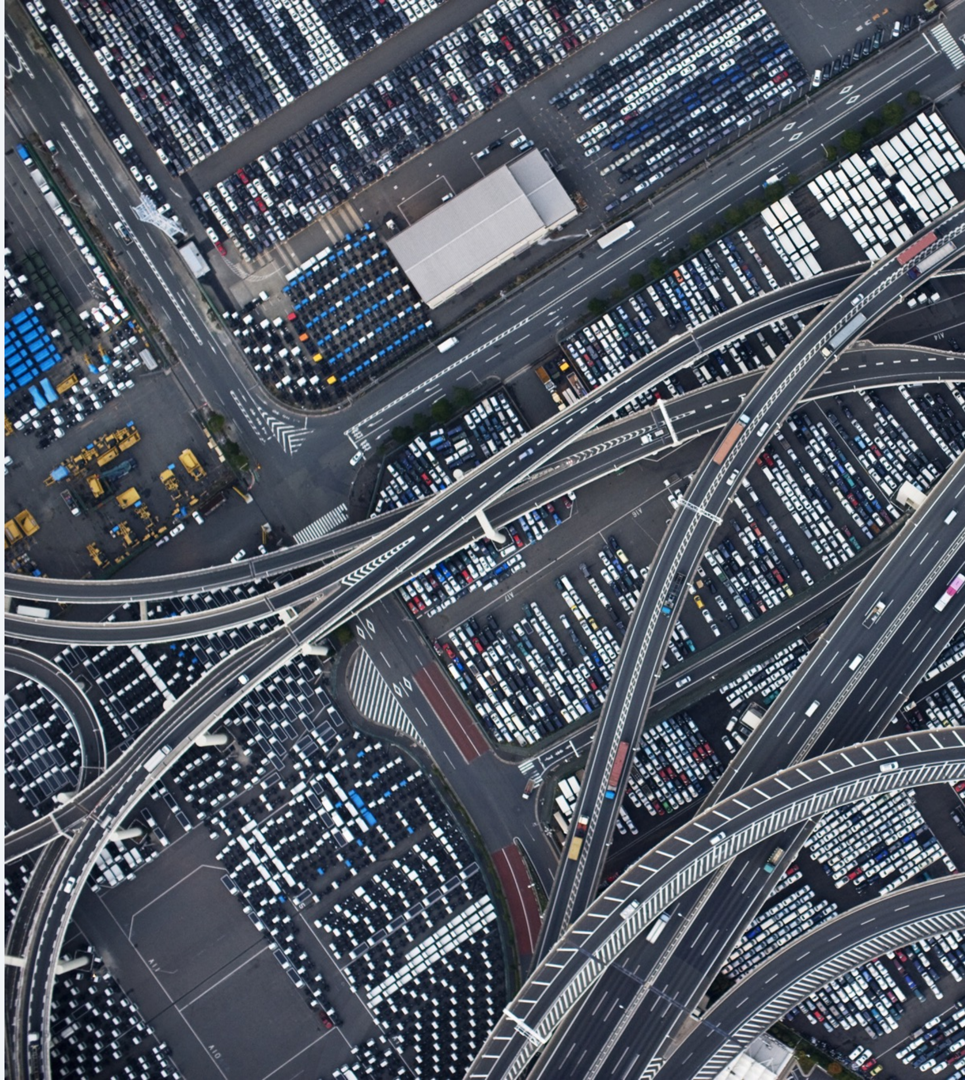
tpf_publish_data –

Transport agnostic api to allow applications to publish a message to guaranteed delivery support (works with MQ, DF + future). Specify a "target" instead of queue / database for configuration-based updates.

Scale away!

On a 4-way TPF system we were able to saturate a 1Gb OSA card without hitting full CPU utilization (30,000 msgs/s at 4KB message sizes using VFA)

Minimal Java overhead - Kafka Producer processing consumes less CPU at steady state than cost to put / read from the DF database.



Conclusion

Use Guaranteed Delivery to provide support today for the infrastructure you need to stay connected to your enterprise architecture.

Update applications to use tpf_publish_data directly or create a custom event dispatcher for integration with business events

Customize support beyond Kafka by writing your own connectors in Java

Thank you!

Let us know if you are interested in adopting this support or we can help you in any way in the path toward adopting Java on z/TPF. For more information contact Daniel Gritter - dgritter@us.ibm.com