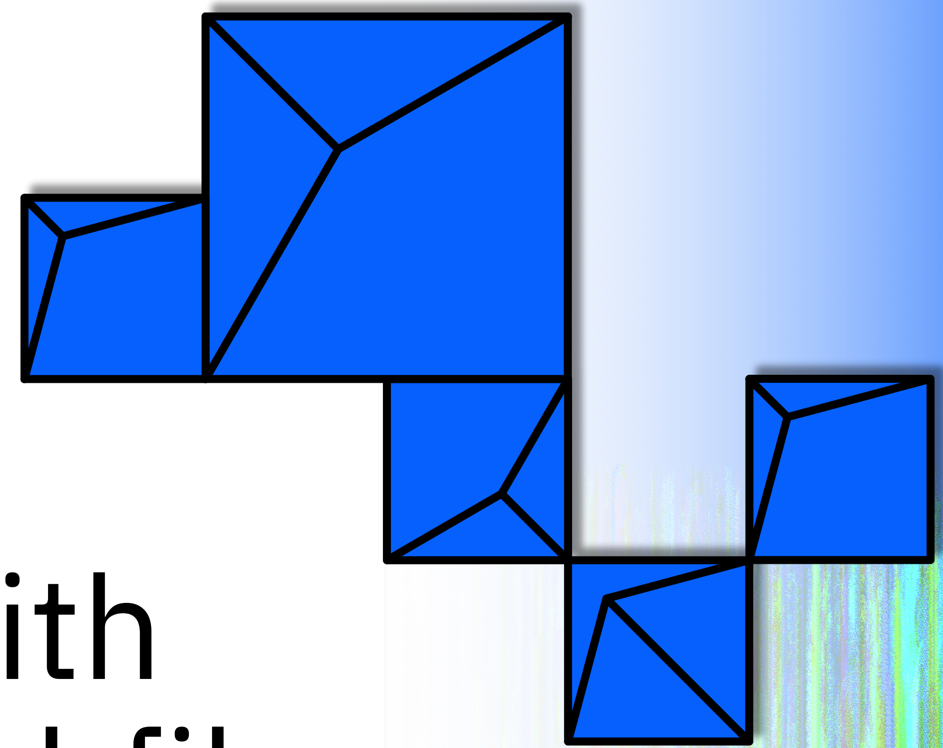
# Data Localization with z/TPFDF Remote Subfiles

Chris Filachek



# Agenda

## Background

- Data localization laws restrict where data is accessed, processed, or stored
    - For example: Data for citizens of “Country.X” must be stored in “Country.X”
    - Any jurisdiction might enact restrictions, including countries or regions
    - Exceptions might exist for processing, cross-border travel, etc.
    - Also referred to as data residency laws

## Background

- z/TPF database architecture provides a database that is:
  - A single, centralized database
  - Always consistent

- Many z/TPF systems store data from customers from around the world.

## Problem Statement

Because of data localization laws, z/TPF applications need to store customer records in different geographic locations.

- Subset of customer records needs to be stored remotely
  - Records can remain in the local database if the data localization laws do not apply to them
- Data localization laws are relatively new over the past few years
  - Expect more countries and regions to adopt these laws and increase the need to store data remotely

## Business Executives

- Enforcement of data localization laws could result in fines or force us to immediately stop doing business in certain countries or regions.

- Data localization laws could prevent us from expanding our business to new countries or regions.

Brian

**Business Executive**

## Application Architects

- We need to store a subset of customer records remotely, but z/TPF does not provide a way to partition data between local and remote storage.

- We could add dual paths to our z/TPF applications, but that would require extensive application updates and significant development time and testing.

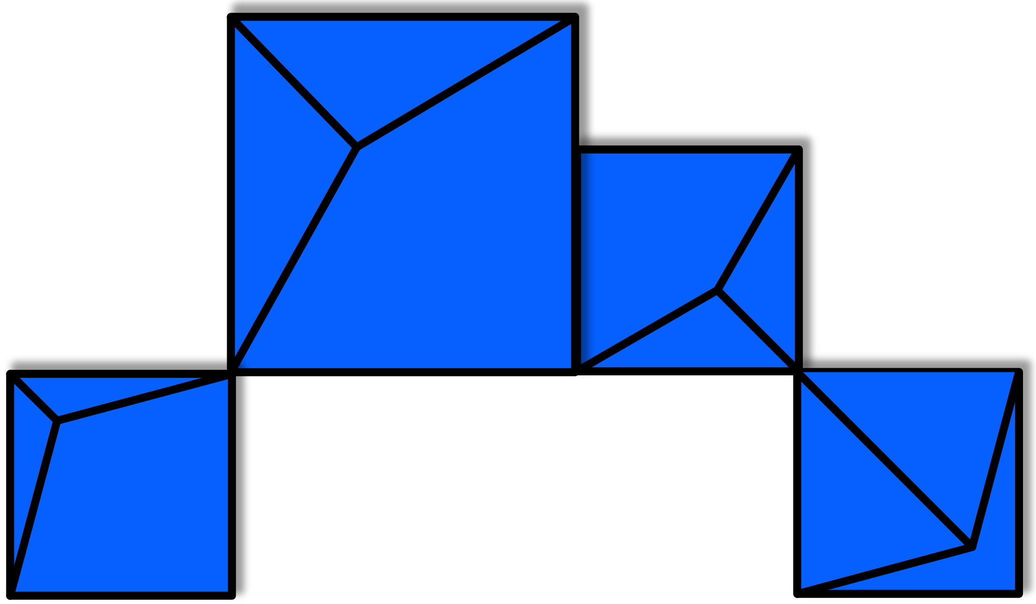
Anna

**Application Architect**

## Value Statement

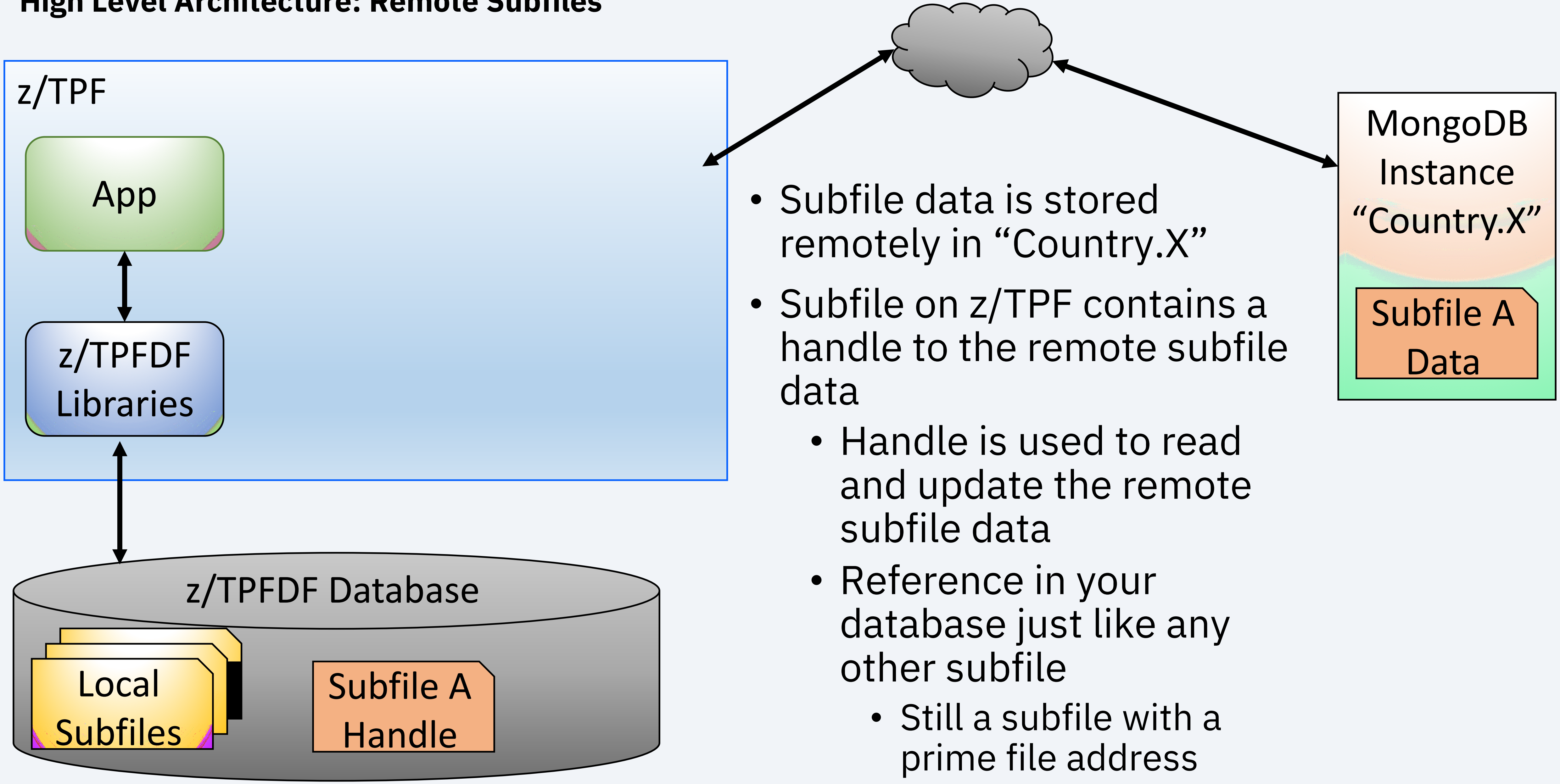
Use z/TPFDF remote subfile support to comply with data localization laws and to grow and maintain business in countries around the world.

- Store individual z/TPFDF subfiles locally or remotely with minimal application changes
- Set up z/TPFDF remote subfile support through user configurable options and without database downtime



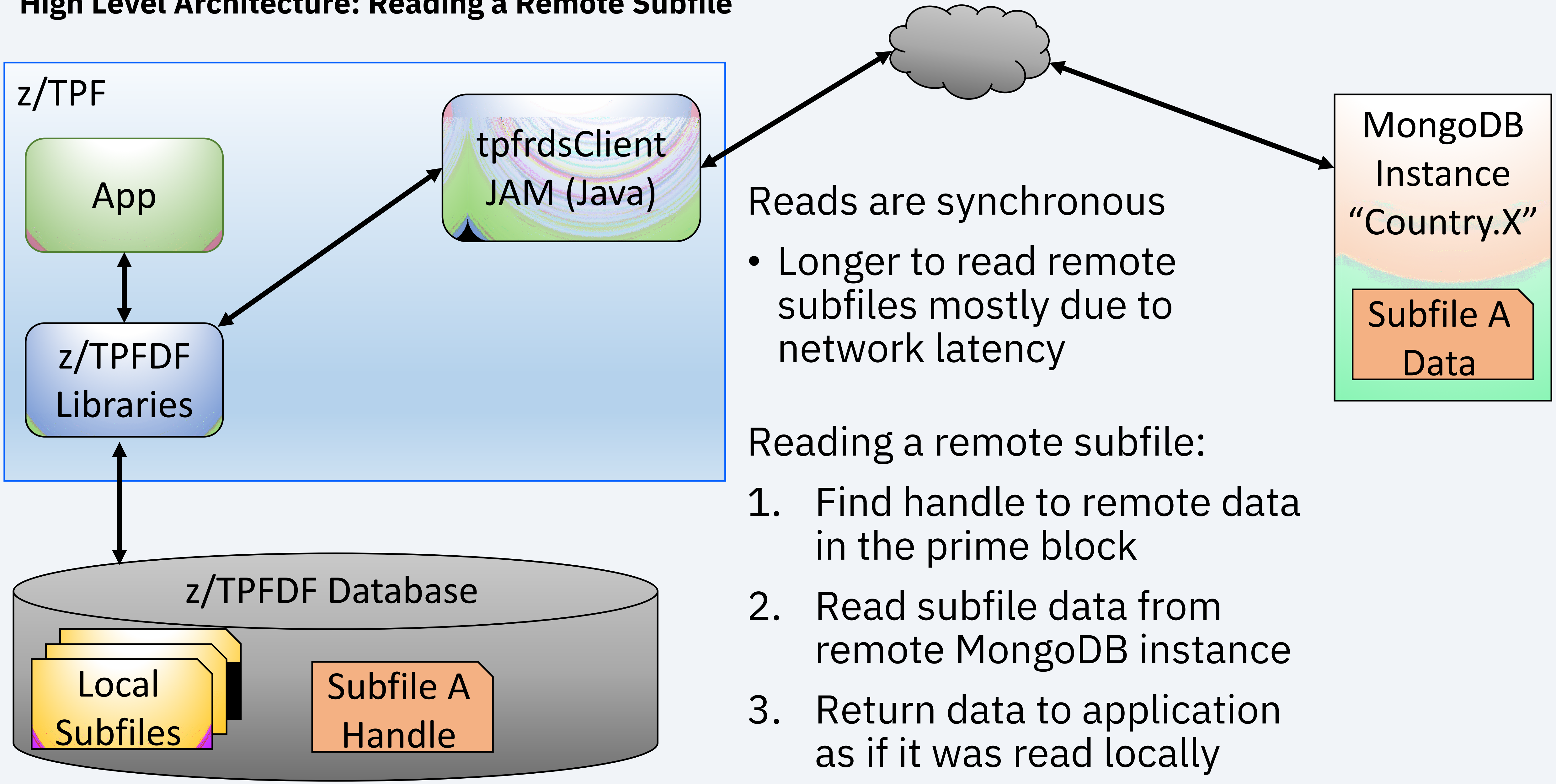
## Remote Subfiles for Architects

## High Level Architecture: Remote Subfiles



- Subfile data is stored remotely in "Country.X"
- Subfile on z/TPF contains a handle to the remote subfile data
  - Handle is used to read and update the remote subfile data
  - Reference in your database just like any other subfile
    - Still a subfile with a prime file address

## High Level Architecture: Reading a Remote Subfile



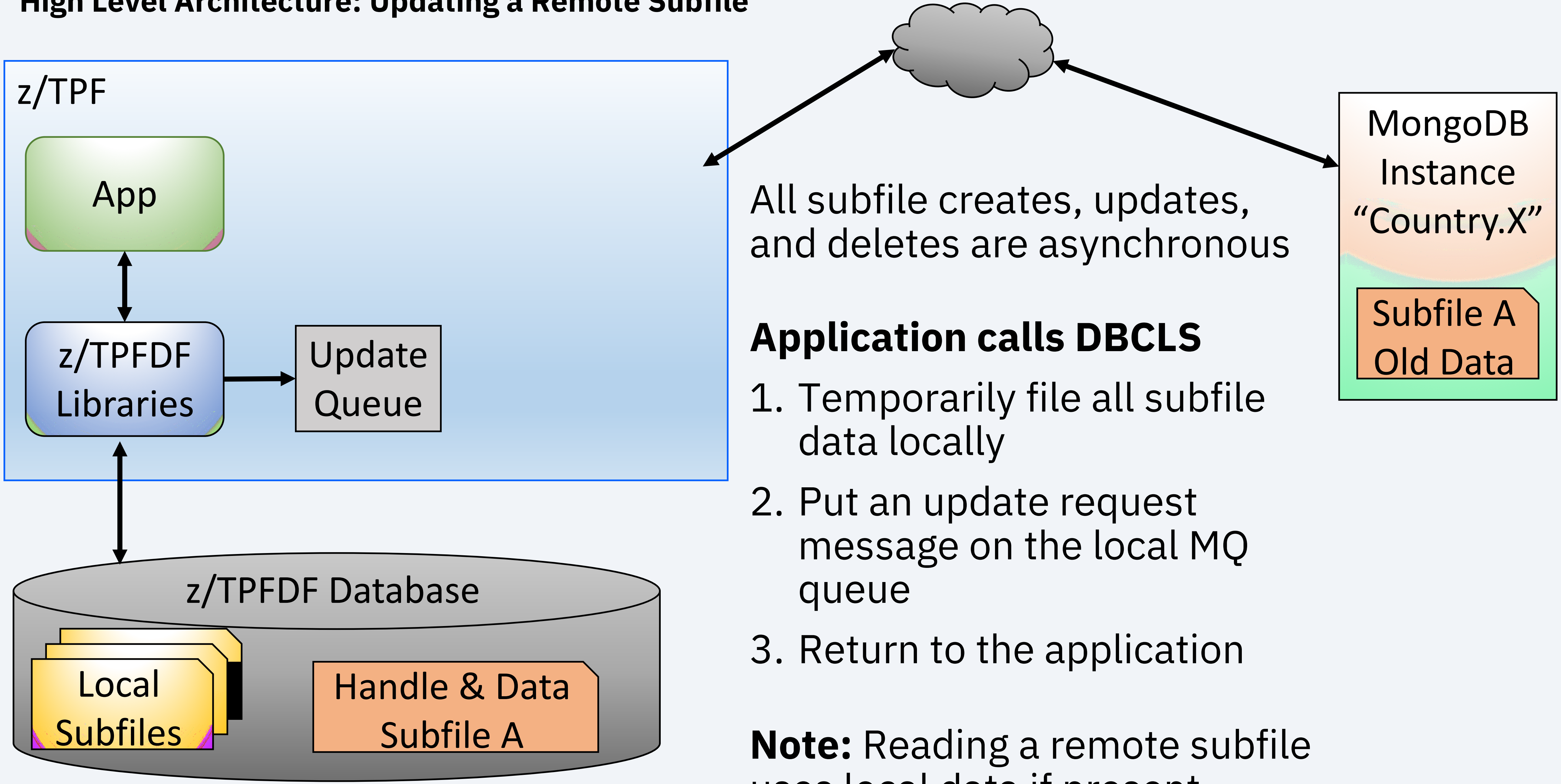
Reads are synchronous

- Longer to read remote subfiles mostly due to network latency

Reading a remote subfile:

1. Find handle to remote data in the prime block
2. Read subfile data from remote MongoDB instance
3. Return data to application as if it was read locally

## High Level Architecture: Updating a Remote Subfile



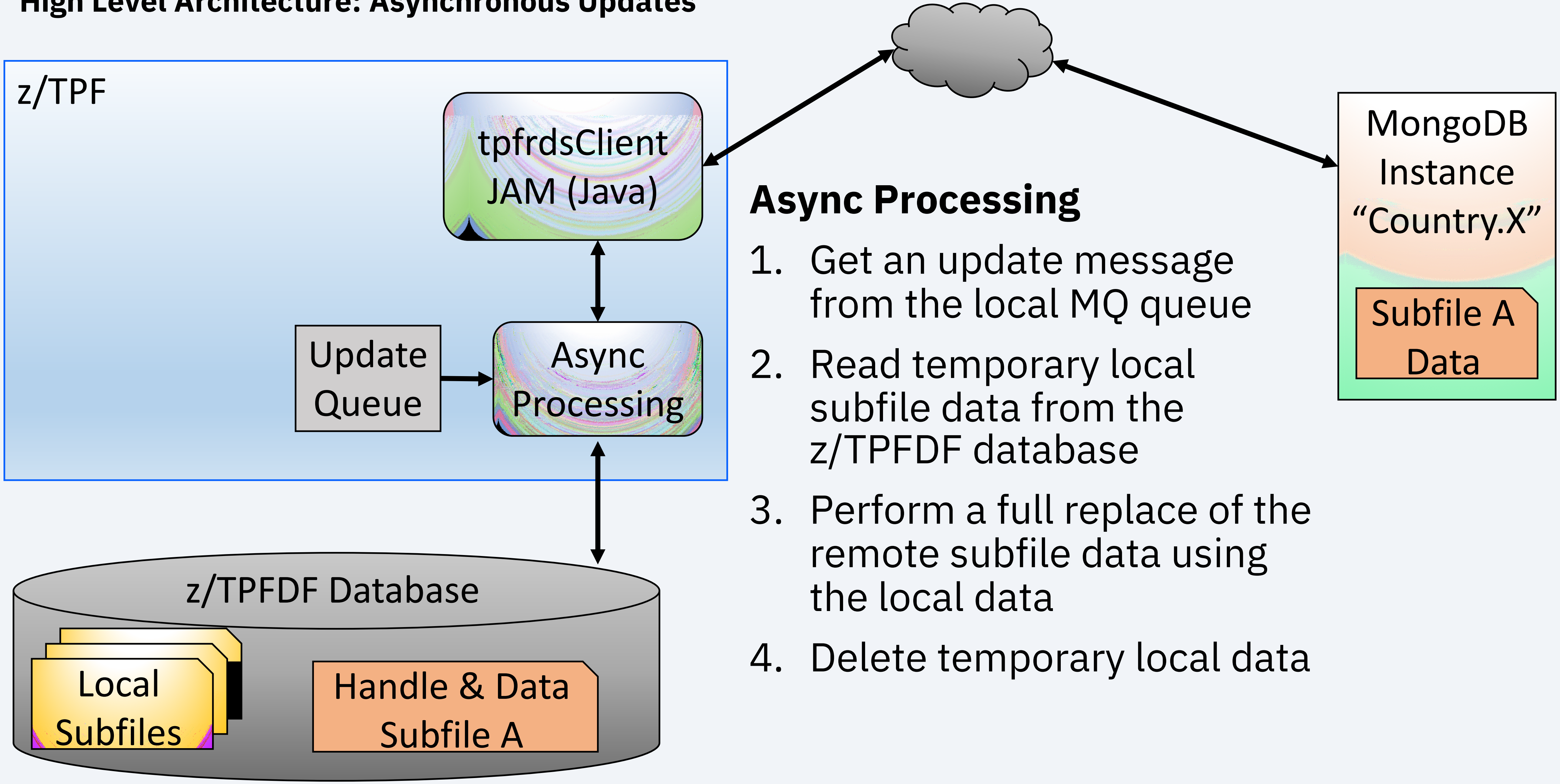
All subfile creates, updates, and deletes are asynchronous

**Application calls DBCLS**

1. Temporarily file all subfile data locally
2. Put an update request message on the local MQ queue
3. Return to the application

**Note:** Reading a remote subfile uses local data if present

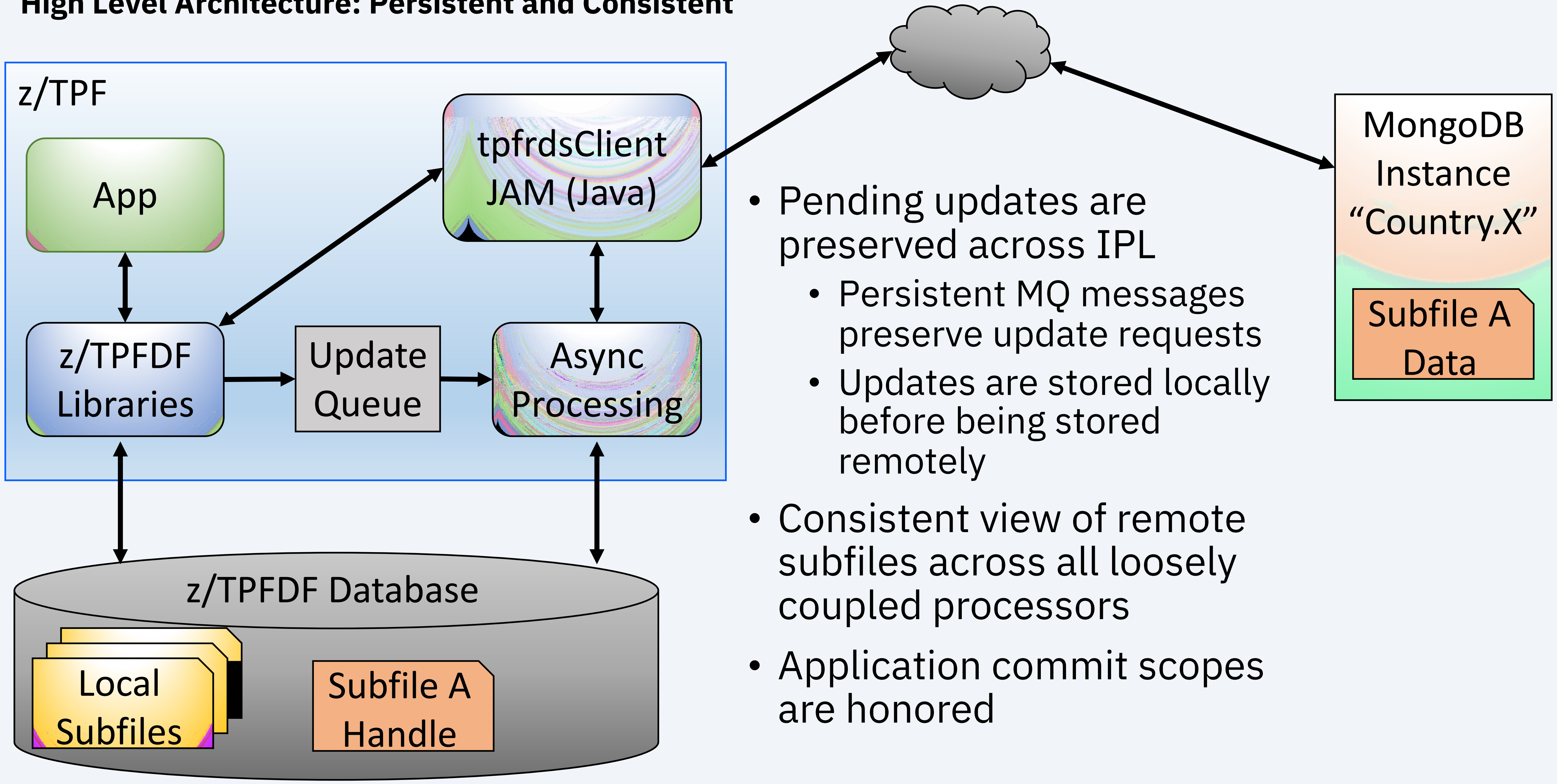
## High Level Architecture: Asynchronous Updates



### Async Processing

1. Get an update message from the local MQ queue
2. Read temporary local subfile data from the z/TPFDF database
3. Perform a full replace of the remote subfile data using the local data
4. Delete temporary local data

## High Level Architecture: Persistent and Consistent

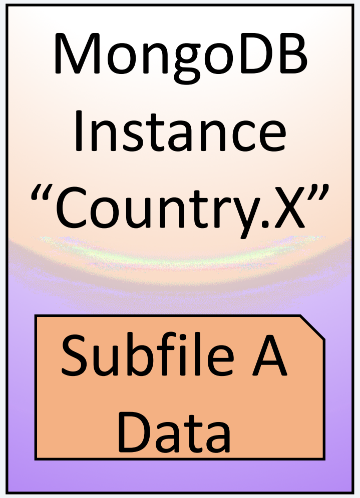


- Pending updates are preserved across IPL
  - Persistent MQ messages preserve update requests
  - Updates are stored locally before being stored remotely
- Consistent view of remote subfiles across all loosely coupled processors
- Application commit scopes are honored

# Remote Subfiles for Database & System Administrators

## Before you begin…

- To use z/TPFDF remote subfile support, your z/TPF system must be configured for z/TPF support for Java™
  - See Configuring your z/TPF system for Java in the z/TPF Knowledge Center for more information
- The tpfrdsClient application manager for Java (JAM) communicates with the remote MongoDB instances
  - Uses the standard MongoDB driver (client) Java package
  - Allows z/TPF to use standard MongoDB connections, operations, and options



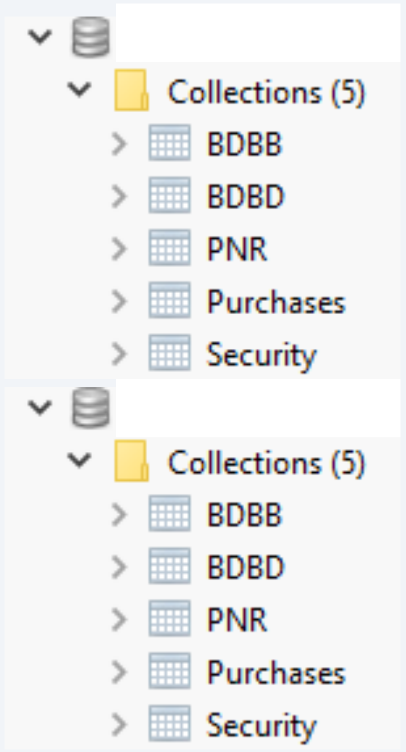
## Remote Data stores

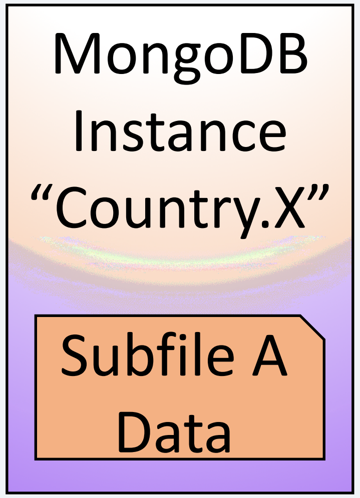
- Each remote data store is a MongoDB instance running on a platform of your choice
  - Configure production remote data stores as high availability clusters
  - Separate remote data stores per country or region

- Remote data store descriptor defines a remote data store to z/TPF (<name>.rds.properties)
  - Defines symbolic name of the remote data store - "Country.X"
  - Contains the MongoDB URI connection string
    - Standard MongoDB URI connection string
    - Define timeout, SSL, authentication, and other connection options

## Configuring a z/TPFDF File for remote subfile

1. Update DBDEF macro to allow remote subfiles
   - DBDEF REMOTEALLOW=YES
   - Allowed for most z/TPFDF files. For example:
     - Must be an R-type file
     - Must use variable length LRECs (no fixed length LRECs)
     - Requires a HOLD on the prime block for updates
     - Can not contain embedded references or use B+Tree support
2. Add the z/TPFDF File ID to a remote file descriptor (<name>.remfil.json)
   - z/TPFDF File ID
   - MongoDB database and collection to use for this file ID
     - Example: Store PNRs in the “PNR” collection under the “TPFCOMPLEX-A” database
   - Optional DFDL schema for this z/TPFDF file





## Updating subfiles in a remote data store

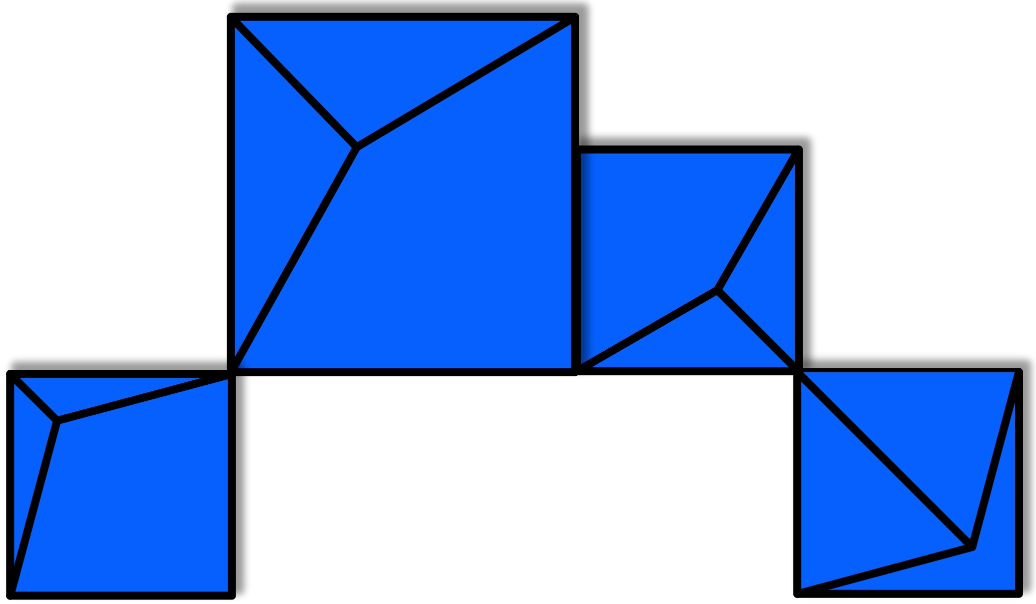
- Only the owning z/TPF complex can update a remote subfile

## Reading subfiles from a remote data store

- Any system can read remote subfiles directly from remote data stores (read-only)
- Customer DFDL schemas allows readers to see formatted subfile data

## Other considerations

- z/TPF complexes can share remote data stores
- z/TPF complexes can not share MongoDB databases and collections
- A collection should contain subfiles for only 1 z/TPFDF file ID



# Remote Subfiles for Application Programmers

## Minimal application changes

- Use the dfrsf_setLocation() API to mark a subfile as remote
  - Set the location using the name of a remote data store defined in a remote data store descriptor
  - Location only needs to be set once for a subfile
    - The subfile is stored in that location until …
      - The subfile is deleted
      - dfrsf_setLocation is called with a different location
- Minimal number of application code paths should need to set the location

## Example

1. dfopn() - Open the subfile
2. Read, add, delete, or modify LRECs
3. If application logic decides that a subfile must be stored remotely…
   a. Determine the name of the remote data store
   b. Set the location for this subfile

      ```
      dfrsf_setLocation(sw00srPtr, “Country.X”);
      ```
4. dfcls() - Close the subfile

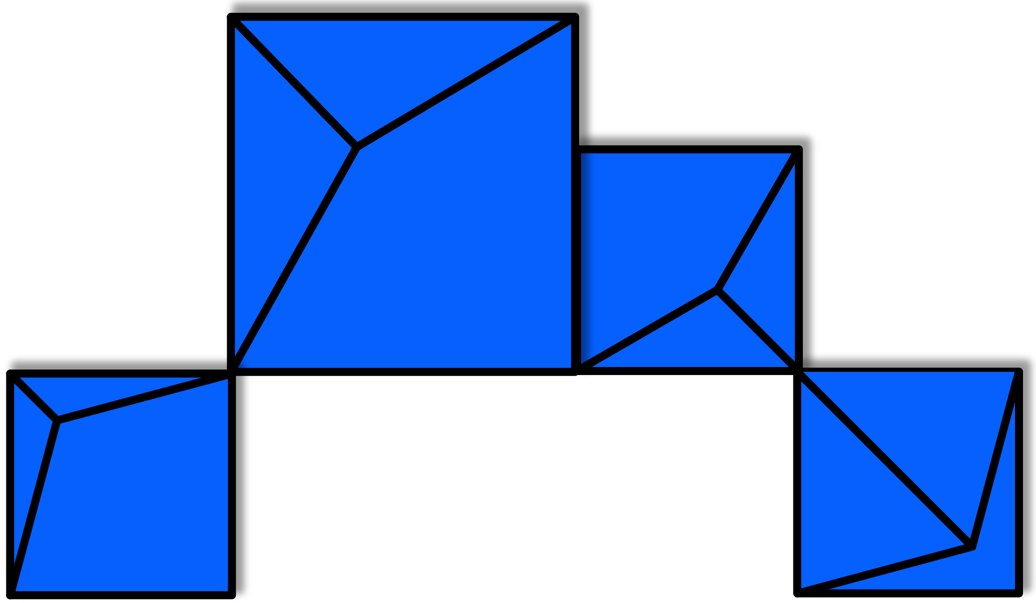
Shortly after the dfcls() completes…

- z/TPFDF asynchronously stores the data in the remote data store
  - Remote subfile data is stored in “Country.X” using the MongoDB database and collection defined for this file ID

## Most applications do not require changes

- No application changes are needed to open or close a remote subfile
  - z/TPFDF automatically manages remote subfiles during open and close processing
  - DETAC mode is forced on when a remote subfile is opened

- No application changes are needed to add, read, update, or delete LRECs for a remote subfile
  - LRECs for a remote subfile are presented to applications as if the subfile was read locally



# Remote Subfiles for Operations and Coverage

## Ease into storing data remotely

- Marking a subfile as remote means…
  - Remote subfile data CAN be stored remotely in “Country.X”
  - If it is stored remotely depends on the remote data store mode for “Country.X”

- Remote data store mode
  - Determines if data for remote subfiles for “Country.X” is stored locally, remotely, or both
  - Slowly transition from storing data locally to storing data remotely
    - Set the mode separately for each remote data store using the ZRDSC command
    - Introduce new remote data stores without immediately storing data remotely

## Remote data store modes

- **LOCAL (default mode)**
  - Data is only stored locally.  Data is read from local copy.
  - z/TPF does not interact with the remote data store.
- **COPY**
  - Data is stored locally and remotely.  Data is read from local copy.
  - z/TPF keeps the remote data updated but does not read from it.
- **VERIFY**
  - Data is stored locally and remotely.  Data is read from remote copy and verified using a SHA-256 message digest.  Fallback to local data on errors.
  - z/TPF uses the remote data but has a local fallback copy if needed.
- **REMOTE**
  - Data is only stored remotely in compliance with data localization laws.

## Managing ECBs

- Reading data from remote data stores
  - Connection issues (timeouts, etc.) can cause a large number of ECBs to wait for read responses
  - Do not want a single remote data store to put your z/TPF system into input list shutdown
  - MAXREADERS is the maximum number of ECBs that are allowed to read from a remote data store at one time
    - ECBs above the MAXREADERS control return an error to the application
    - Set separately for each remote data store using the ZRDSC command

- Updating data in remote data stores
  - Updates are processed asynchronously by IBM controlled ECBs
  - Maximum number of IBM controlled ECBs limited by MAXWRITERS
    - Set separately for each remote data store using the ZRDSC command

## Monitoring z/TPFDF remote subfile support

- Metrics for each remote data store
  - Rate of read, create, update, and delete operations
  - Error and timeout counts
  - Average and highwater number of reader and writer ECBs
  - Included in data collection/reduction and continuous data collection (CDC)

- Remote subfile counters in z/TPFDF statistics
  - Remote read, create, update and delete counts per z/TPFDF file ID
  - Included in data collection/reduction, continuous data collection (CDC), and name-value pair collection

- Console messages
  - Issue throttled messages to the console for issues related to accessing remote data stores (timeouts, authentication errors, etc.)

## Recoup

- Recoup does not read the data from remote data stores
  - Chain chase only reads local z/TPF records
- Recoup runtime should not be impacted

## CRUISE

- CRUISE supports a new RSF option
  - Applies to VERIFY, PACK, CAPTURE, and RESTORE functions
- RSF=NO only processes data stored locally on z/TPF
- RSF=YES processes both local data and remote data stored in remote data stores

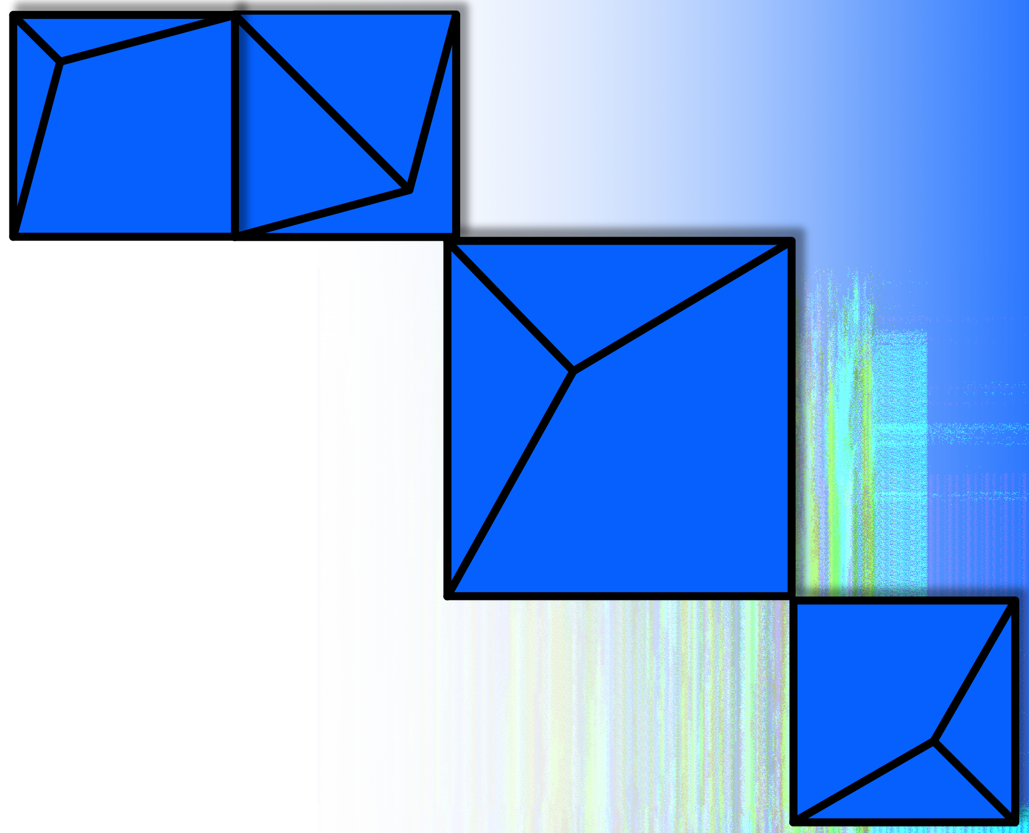
Conclusion

## z/TPFDF Remote Subfile support

- APARs PJ45756 and PH11394
- z/TPF Level 2020  (June 2020)

# Thank You

Questions? Comments?



## Virtual TPFUG Q&A

Summary of Q&A from the virtual TPFUG event:

| Question | Answer |
|---|---|
| | |
| | |
| | |

## Virtual TPFUG Q&A

Summary of Q&A from the virtual TPFUG event:

| Question | Answer |
|---|---|
| | |
| | |
| | `ZRDSC SET NAME-CountryA MAXWRITERS-10` |

## Virtual TPFUG Q&A

Summary of Q&A from the virtual TPFUG event:

| Question | Answer |
|---|---|
| | |
| | |

## Virtual TPFUG Q&A

| Question | Answer |
|---|---|
| | |

# Trademarks