z/TPFDF Performance and Storage Improvement Update

Database / TPFDF Subcommittee

Chris Filachek
IBM z/TPF Database & Storage Architect

2025 TPF Users Group Conference
May 5-7, Austin, TX

IBM Z

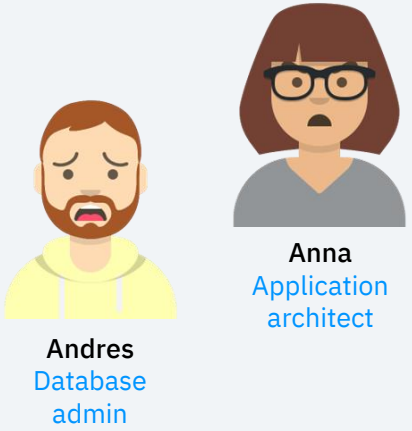
IBM

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.

Problem Statement

The accumulated service time from reading long overflow chains can **negatively impact application response times and exceed SLAs.**



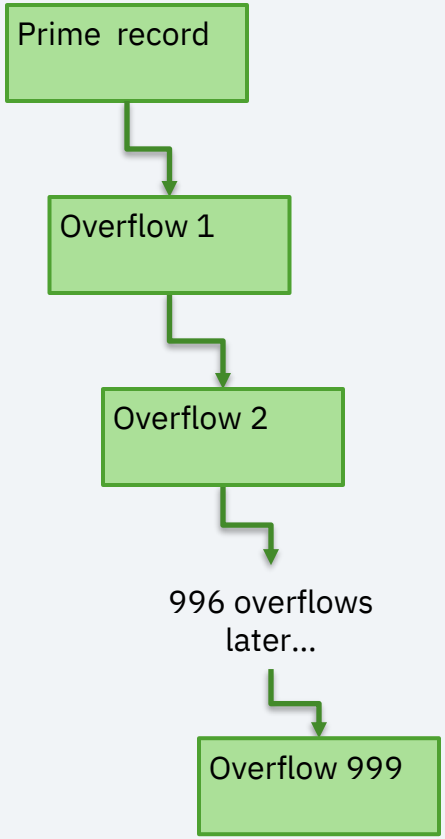
Andres
Database admin

Anna
Application architect

Our applications need to store more data to meet business requirements, but reading long chains is adding 250 milliseconds or more to the application response time.

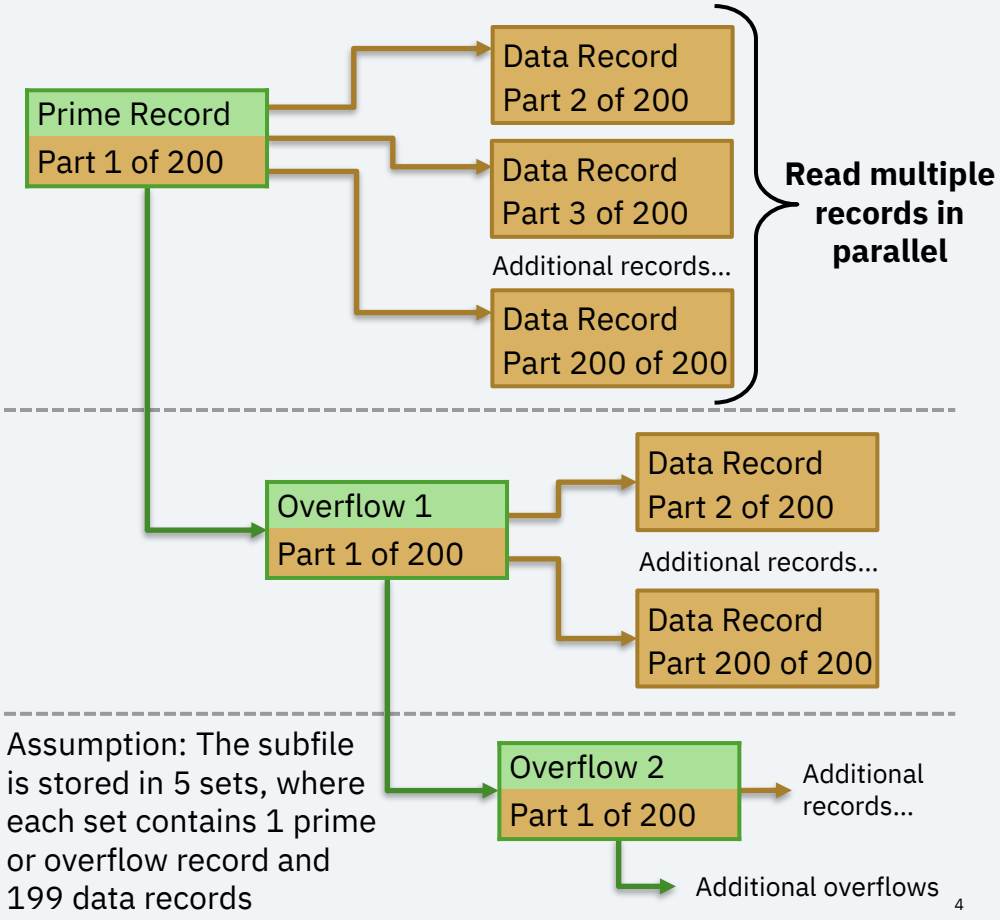
We're not sure how to address business requirements and meet SLAs at the same time.

Standard z/TPFDF subfile with 1000 records in chain



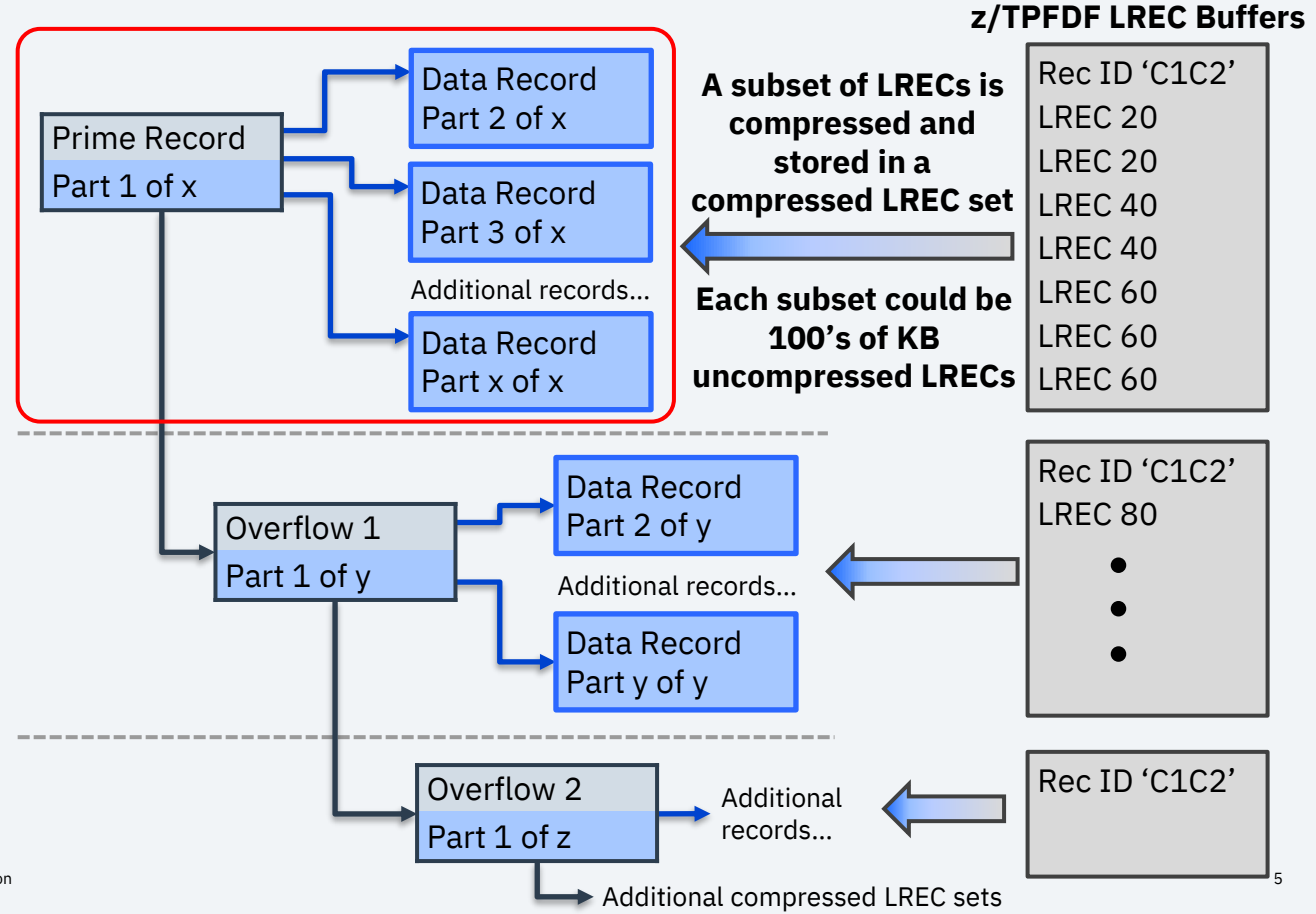
What If We Could Read Records in Parallel?

- Instead of a long overflow chain, chain a subset of records to each prime or overflow record
- Reading all LRECs in a standard subfile with 1000 records requires **1000 serial read I/O requests**
- By reading records in parallel, those same records can be **read in 6 batches** of up to 200 records at a time
  - Read each set of data records and next overflow in parallel
  - Reading records in parallel significantly reduces accumulated service time
  - **However, the subfile is still stored in 1000 physical records on DASD!**



Compress to Reduce Storage and Response Times!

- A **compressed LREC set** is a prime or overflow record and 0 or more 4 KB data records
  - Read data records in parallel
  - Store the same LRECS in fewer physical records
- **Reduce accumulated response time AND storage needs!**
- For more details, see the 2024 TPFUG presentation



Value Statement

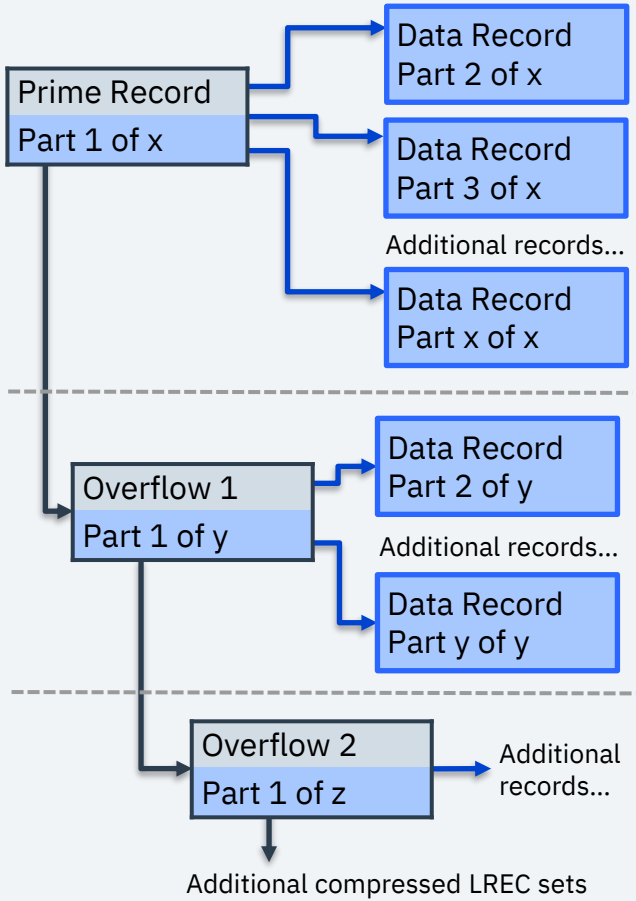
Applications that need to read z/TPFDF subfiles with long overflow chains can **read the same amount of data in a fraction of the time without requiring application changes or database downtime.**

Andres
Database admin

Anna
Application architect

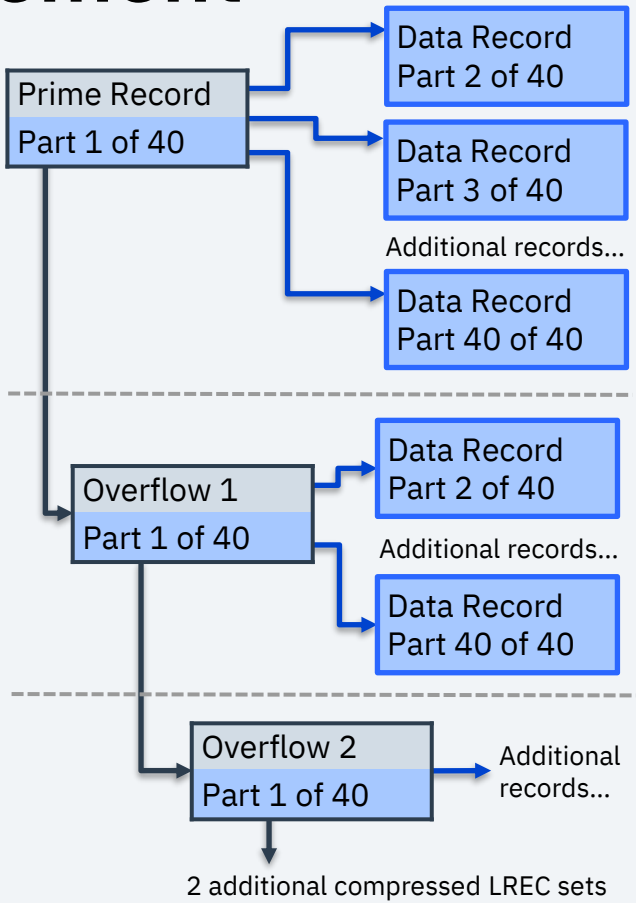
By migrating to compressed z/TPFDF subfiles, we can store additional data in support of our business requirements while still meeting application response time SLAs.

Compressed z/TPFDF Subfile



Example Service Time Improvement

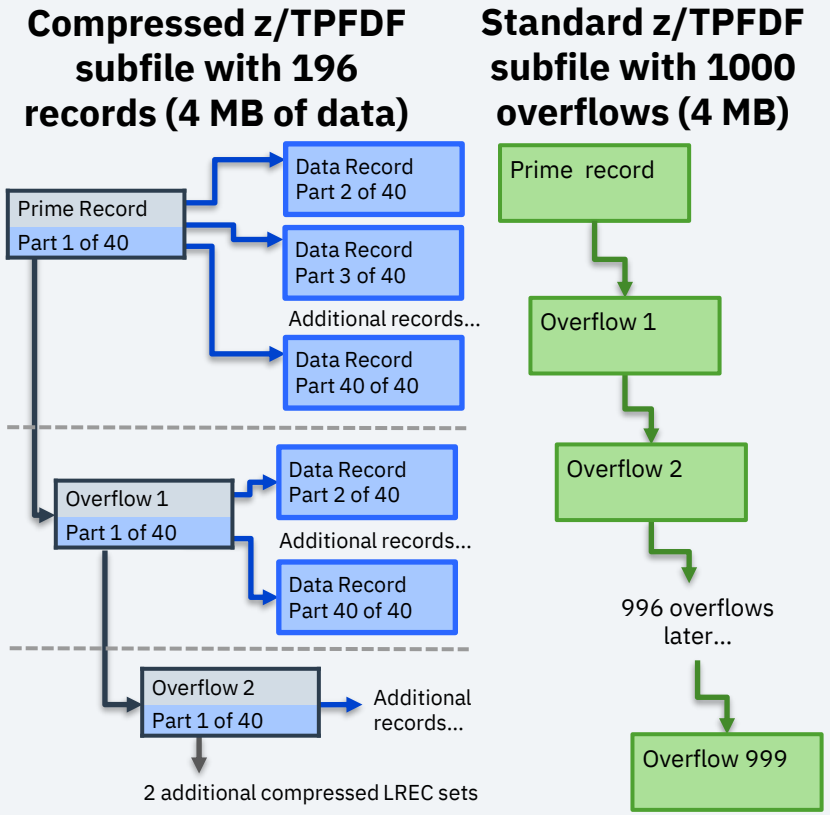
- An uncompressed, standard 4 MB subfile can be read from DASD in **250 ms**
  - Assumes 1000 records and 0.250 ms service time
- A compressed subfile that contains 4 MB of uncompressed data could be read from DASD in **under 3 ms**
  - **99% reduction in accumulated service time**
  - **80% reduction in storage needs (only 196 records)**
    - 5 compressed LREC sets (prime and 4 overflows)
    - Sets 1-4 contain 39 data records, set 5 has 35
  - Read each set of data records and next overflow record in parallel
  - Assumes 80% compression ratio and 0.250 ms service time



Access Patterns

Access patterns are critical to understanding what design choices and configuration options are required to provide the right solution to meet your needs

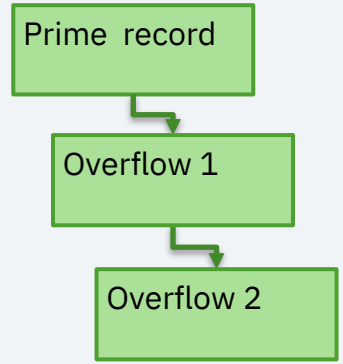
- Example 1: Read full subfile and no updates
  - Standard: 1000 serial read I/O requests
  - Parallel & Compressed: 6 batches of up to 40 read I/O requests each (196 I/O requests)
  - **Less I/O and less accumulated response time**
- Example 2: Read and update an LREC in the prime record
  - Standard: 1 read & 1 write (2 I/O requests)
  - Parallel & Compressed: 3 batches of up to 40 I/O requests each (81 total I/O requests)
  - **More I/O and more accumulated response time, therefore not a good candidate for compression**



1) Read prime
2) Read 39 data records and 1st overflow
3) Write prime and 39 data records

Problem Statement

Millions of standard z/TPFDF subfiles with only a few records in chain



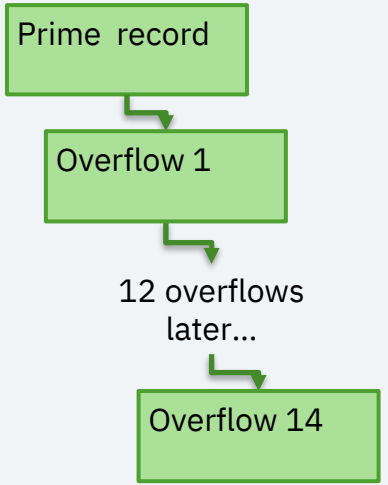
Database growth is depleting the number of available pools faster than projected, requiring the business to **add physical storage or refresh DASD.**

Andres
Database admin

Calvin
Capacity planner

Business growth and regulatory requirements are causing the number of z/TPFDF subfiles to grow faster than expected.

Physical storage and new pool records must be added before available pools fall below critical levels.



Value Statement

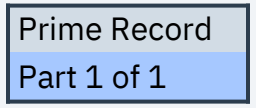
Your z/TPFDF databases can support growth by storing z/TPFDF subfiles with overflow records **in less space and without requiring application changes, database downtime, or more DASD storage.**

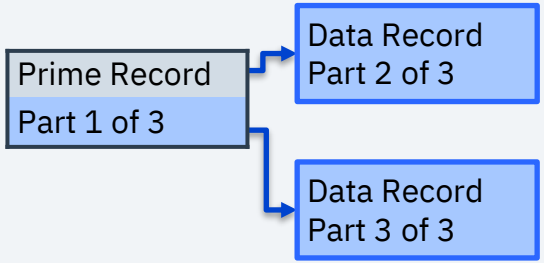
Andres
Database admin

Calvin
Capacity planner

By migrating to compressed z/TPFDF subfiles, we can support business growth and regulatory requirements without requiring additional DASD storage.

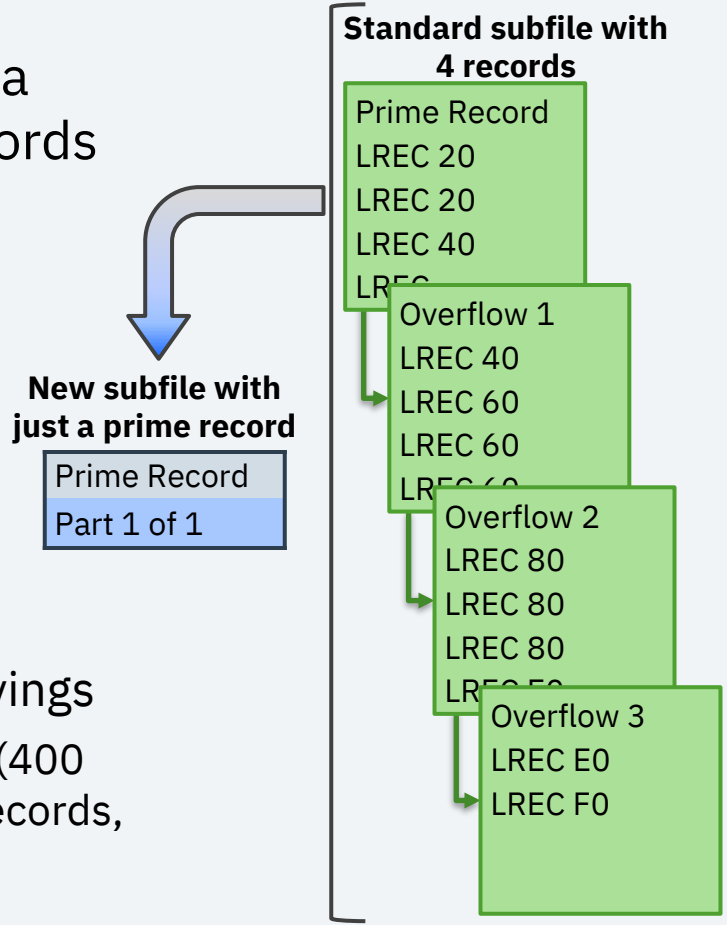
Millions of z/TPFDF subfiles, where each subfile is compressed to a single record or only a few records





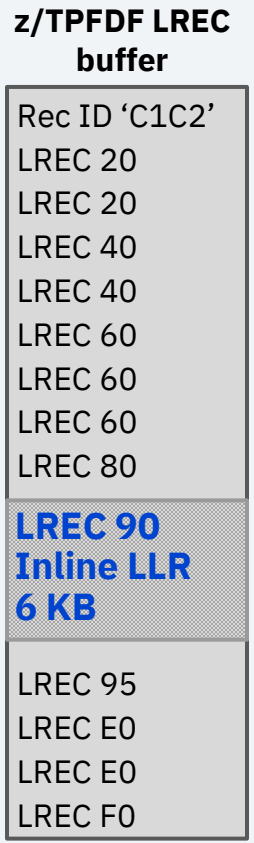
Savings for Subfiles with a Few Overflows

- A standard subfile with 14 KB of LREC data could be stored in as few as four 4 KB records
- With a compressed subfile and assuming 80% compression…
  - Store the same data in a single record: **75% reduction in the number of records**
  - Read the same subfile in a single I/O: **75% reduction in total service time**
- Databases with millions of subfiles and a few overflows per subfile could see substantial savings
  - Example: 100 million subfiles of 4 records each (400 million records) could be stored in 100 million records, saving 300 million records



An LLR is Just Another LREC

- Large logical LRECs (LLRs) are larger than a core block (4 KB) and stored in 2 or more records
  - Buffered access mode (BAM) and additional DASD I/O are required to read and update an LLR
- Compressed LREC sets are uncompressed into z/TPFDF LREC buffers in ECB heap
  - A buffer could be 100s of KBs in size and could hold LRECs larger than 4 KB
- Plan to support “inline LLRs” with compressed subfiles
  - Inline LLR is an LLR up to 32 KB in size
  - Stored “inline” with other LRECs
  - Compressed and stored with standard LRECs
  - Might be able to access without BAM



Value Statement

Smarter applications can make better business decisions by **accessing significantly more data with no additional latency.**

With the ability to store more data and quickly access that data, what does that enable?

- Optimize decision making for your business by accessing more data per transaction
- Improve your customers' experience by returning more information or providing more options
- Save more information per transaction to use as input into AI and analytics

Requirements

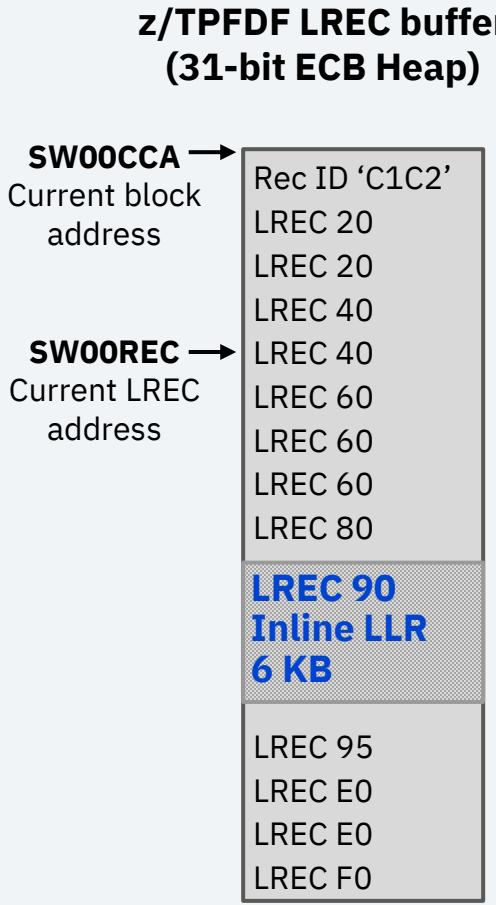
- The z/TPFDF file must meet the following requirements:
  - Defined as a z/TPFDF R-type file
  - Not defined as a B+Tree data or node file
  - Not defined as a block index file
  - Not using algorithm #TPFDB0D
- An IBM z15 or later is required for on-chip hardware compression

Assumptions

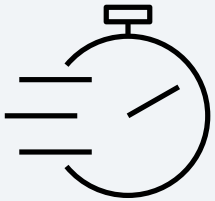
- Supported with…
  - z/TPFDF encryption – **Compress AND encrypt your subfiles!**
  - Data events for z/TPFDF – **No changes to formats, transmit protocols, data event user code, or event consumers!**
  - Other z/TPFDF features will be supported based on the needs of sponsor users
- Enabled and controlled through DBDEF parameters
- Migrate existing databases by using CRUISE PACK or similar functions

Application Considerations

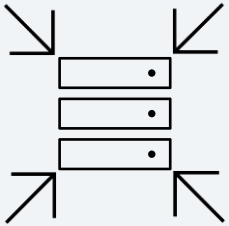
- Applications that use only z/TPFDF APIs to access subfiles and LRECs should not require any changes
  - Support most z/TPFDF APIs (read, add, replace, delete, open, close, checkpoint, and more) and most options
  - Special z/TPFDF APIs like sort, merge, copy, restore or the z/TPFDF tape APIs will not be supported with the initial release
- Application accessible SW00SR fields, **SW00CCA and SW00REC**, that normally contain core block addresses will contain 31-bit ECB heap addresses for compressed subfiles
  - This change should not affect applications
- Application accessible SW00SR field **SW00NAB** (next available byte) is 2 bytes and can't accommodate buffers larger than 64 KB
  - Plan to support the existing 2-byte and a new 4-byte SW00NAB field
  - Only applications accessing SW00NAB for compressed subfiles would need to be updated to use the new 4-byte SW00NAB field



Summary - Value Statements



Applications that need to read a significant amount of data from existing or new z/TPFDF subfiles with long overflow chains can **read the same amount of data in a fraction of the time without requiring application changes.**



Your z/TPFDF databases can support growth from normal business expansion and new requirements by storing z/TPFDF subfiles with overflow records **in less space and without requiring application changes.**

Smarter applications can make better business decisions by **accessing significantly more data with no additional latency.**

Be a sponsor user

Sponsor users assist in design and implementation, and your feedback drives our development cycle.

To design this support, we need to understand YOUR z/TPFDF databases:

- Large or highly accessed files
- Large number of subfiles, each with some overflows
- Large logical records (LLRs)
- Access and update patterns

Target personas

- Application architects
- Database administrators

Sponsor user meetings are in progress

Interested? Contact

Chris Filachek (filachek@us.ibm.com)

Thank you