z/TPFDF Performance and Storage Improvements

Database / TPFDF Subcommittee

Chris Filachek
IBM z/TPF Database & Storage Architect

2024 TPF Users Group Conference
May 5-8, New Orleans, LA

IBM Z

IBM

Disclaimer

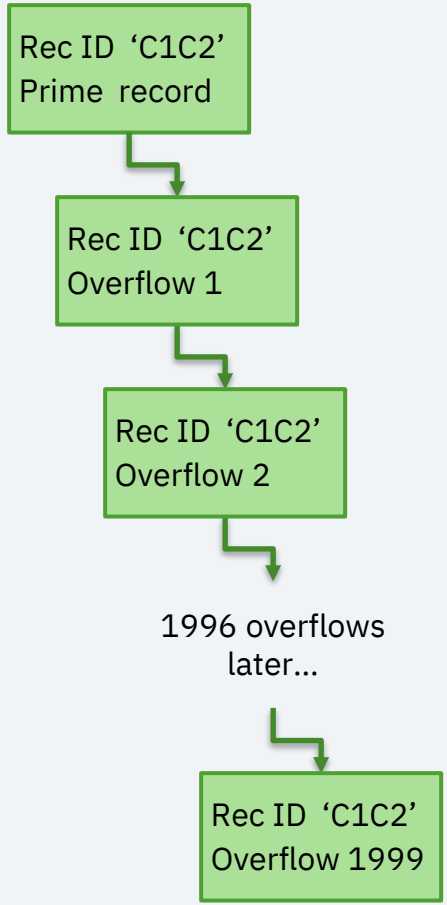
Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Problem Statement

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

- Exceed SLAs before even accounting for processing time
- Even with average DASD service times around 0.250 ms, the accumulated service time to read a subfile with a long overflow chain could approach a ½ second or more

Standard z/TPFDF subfile with 2000 records in chain



As-Is User Story

Carol is investigating why the response time for one of their applications is exceeding their SLAs.

She starts by using runtime metrics collection to look at a message type associated with this application.

By using correlation analysis in runtime metrics collection, Carol discovers that the number of overflow record finds for a z/TPFDF record ID has been steadily increasing over the past several months.

Digging deeper, Carol reviews the recoup statistics for this record ID and finds the average chain length per subfile has also been increasing during this time.

Carol
Coverage programmer

As-Is User Story

Carol discusses this with Anna and Andres and learns that business requirements have increased the amount of data stored in each subfile.

New application logic reads this additional data, increasing the amount of I/O and accumulated service time for this application and message type.

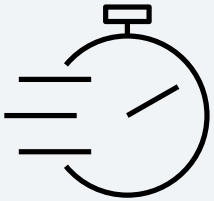
To meet application response time SLAs, they are afraid they might have to change the application to process less data, which might result in less than optimal business decisions.

Carol
Coverage programmer

Anna
Application architect

Andres
Database admin

Value Statement



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 data in a fraction of the time without requiring application changes.**

To-Be User Story

Carol
Coverage programmer

Anna
Application architect

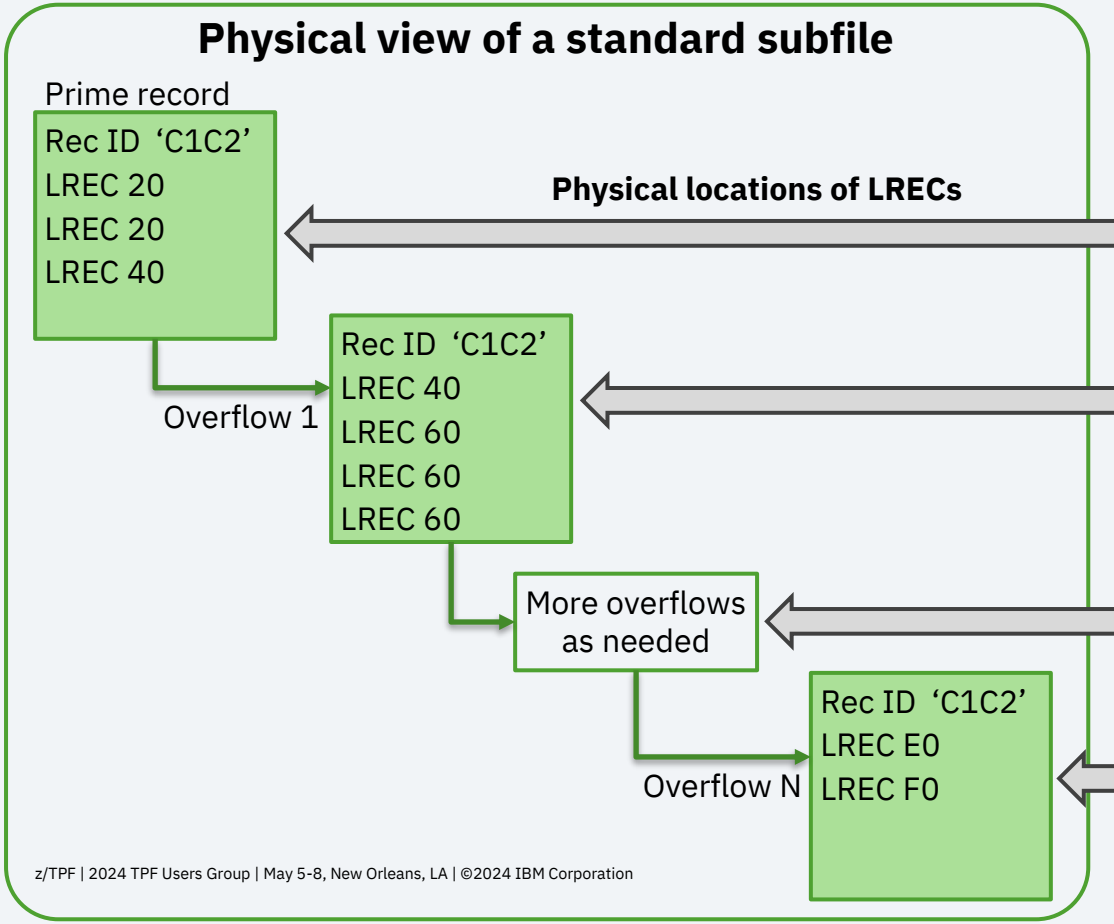
Andres
Database admin

Carol is investigating why the response time for one of their applications is exceeding their SLAs.

Anna tells her that business requirements have increased the amount of data stored and processed per subfile.

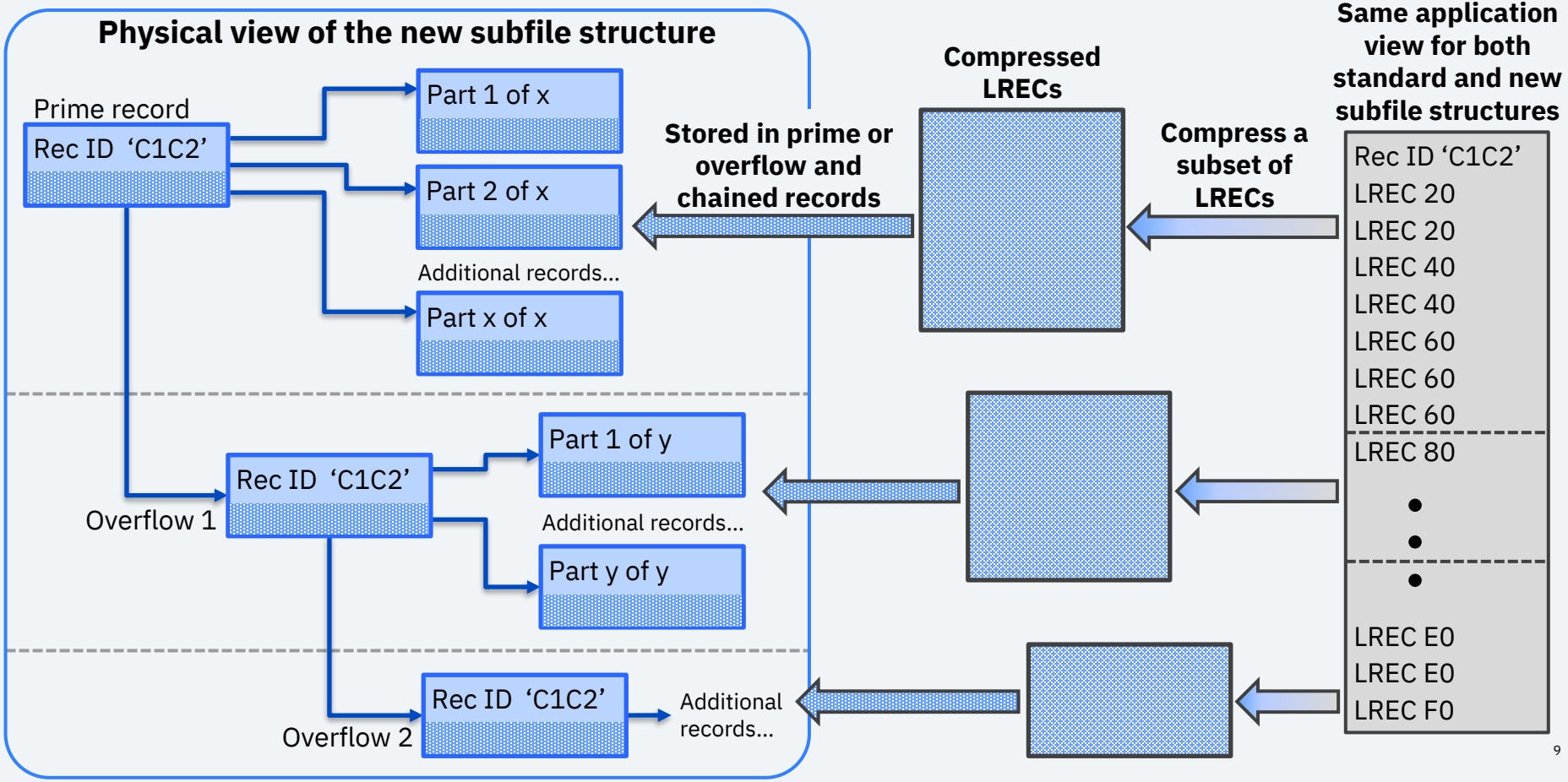
Andres informs them that migrating to a new z/TPFDF subfile structure could help them meet SLAs by **significantly reducing the amount of time to read the same amount of data, without requiring application changes or database downtime.**

Background: Standard z/TPFDF Subfiles



- When using z/TPFDF APIs to access LRECs, applications do not know which LRECs are in which records
- Applications view a subfile simply as a series of LRECs

Rethinking the z/TPFDF Subfile Structure

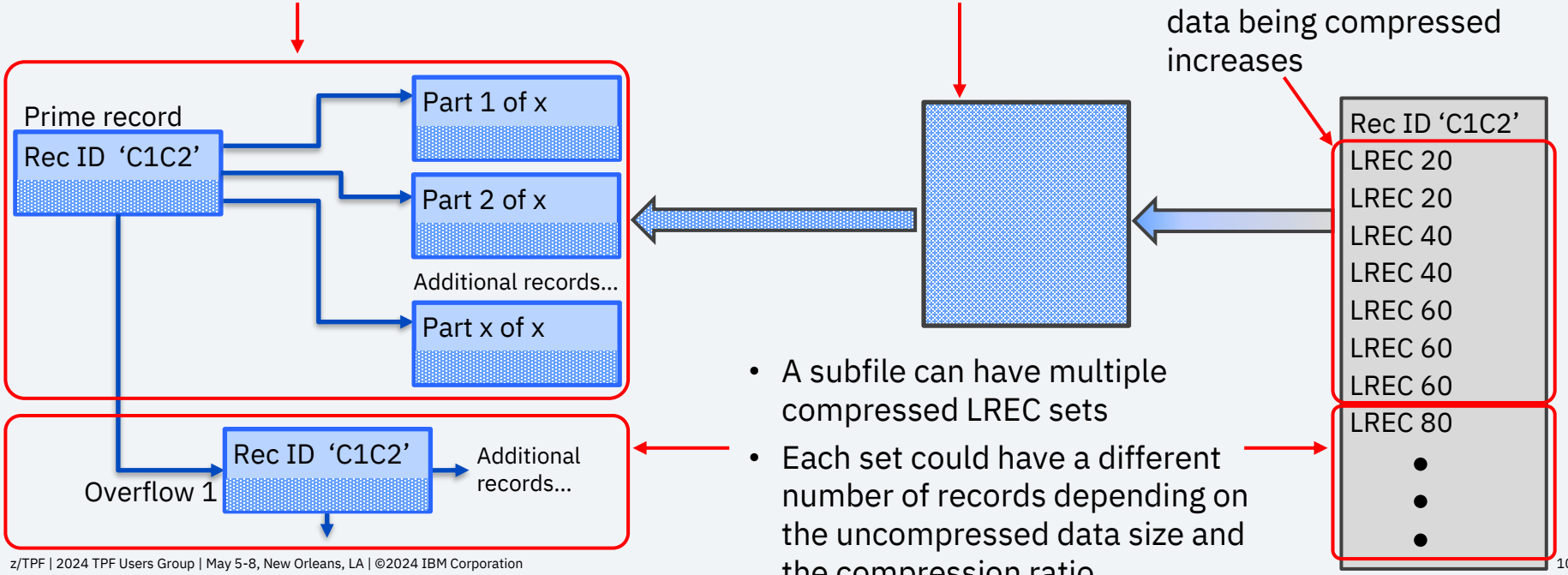


New Subfile Structure Considerations

- A compressed LREC set is the set of records and compressed data for 1 set of LRECs
- Compressed data is stored in a prime or overflow record and chained 4 KB records

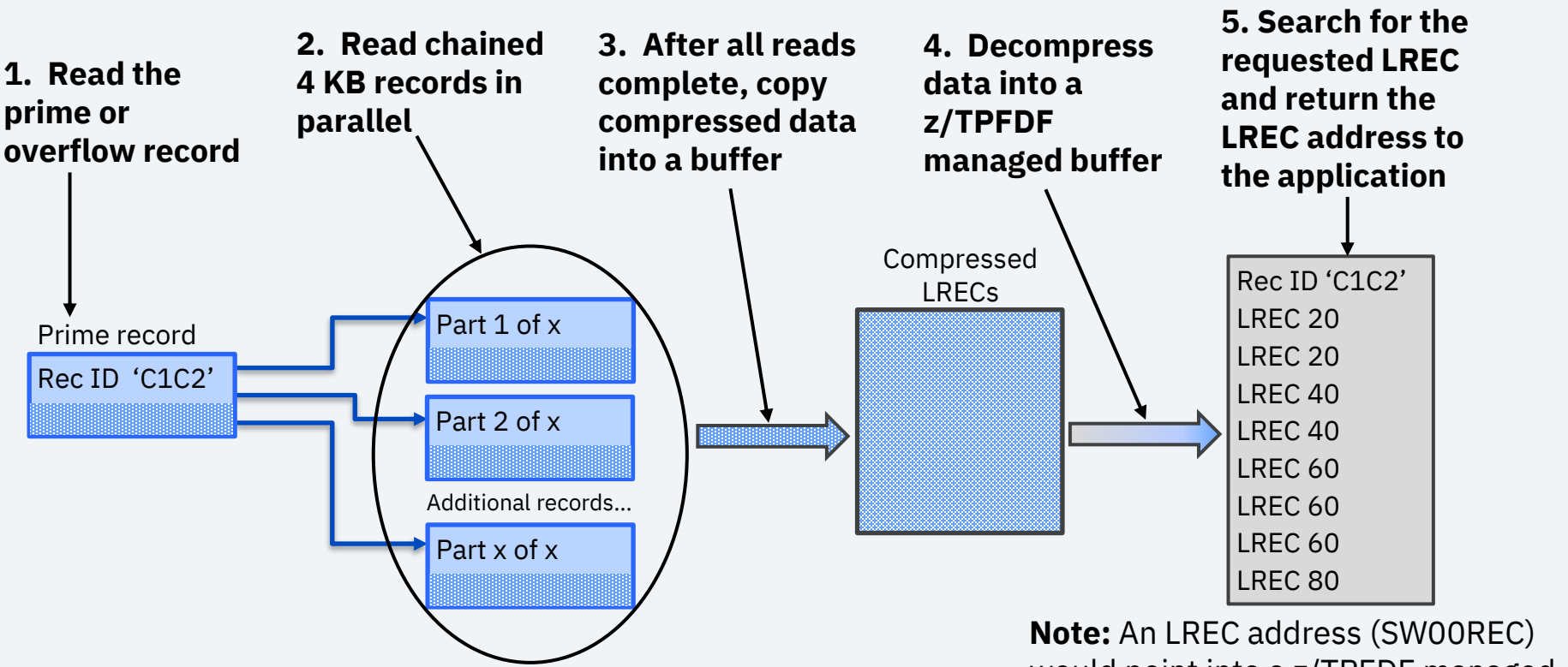
- Size of compressed data could be more than 4 KB and stored in multiple records

- Compress a contiguous set of LRECs, where each set could be 100's of KB of data
- Compression is more effective as the amount of data being compressed increases



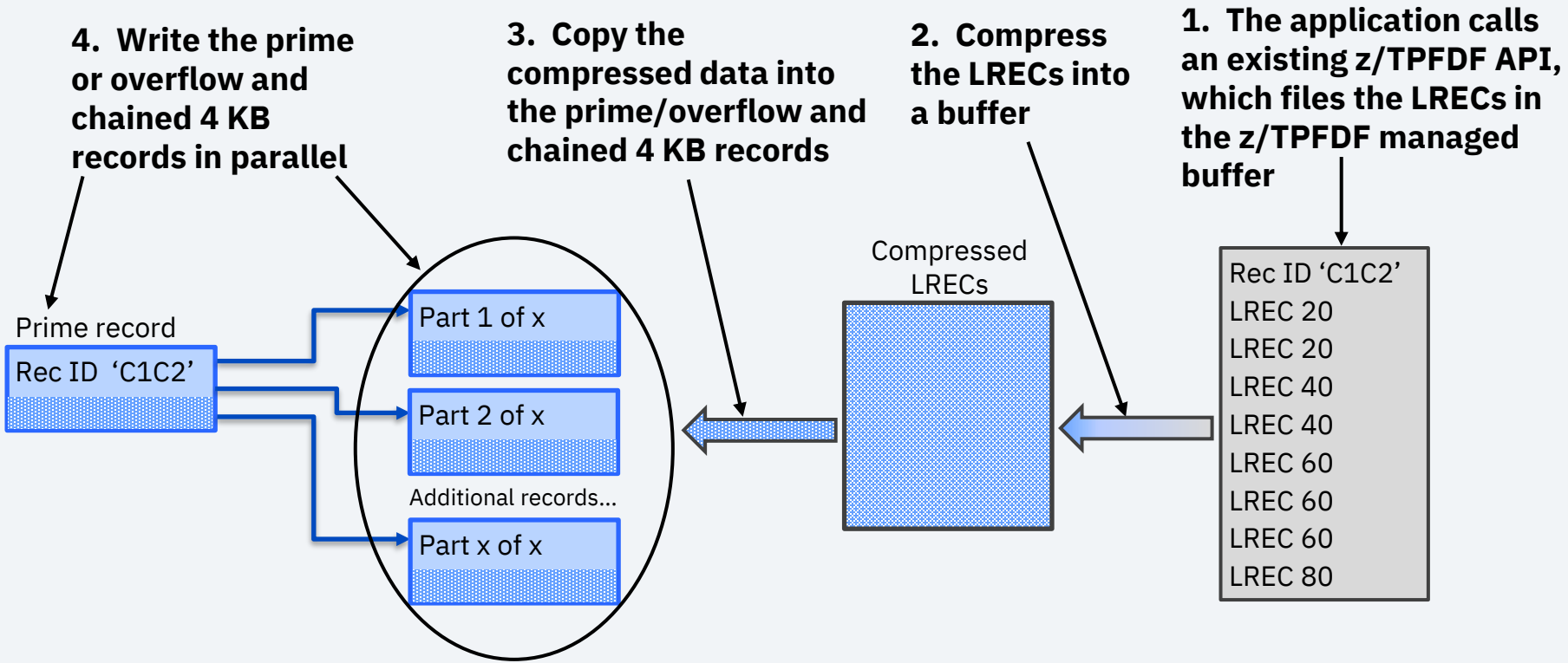
- A subfile can have multiple compressed LREC sets
- Each set could have a different number of records depending on the uncompressed data size and the compression ratio

Reading a Compressed LREC Set



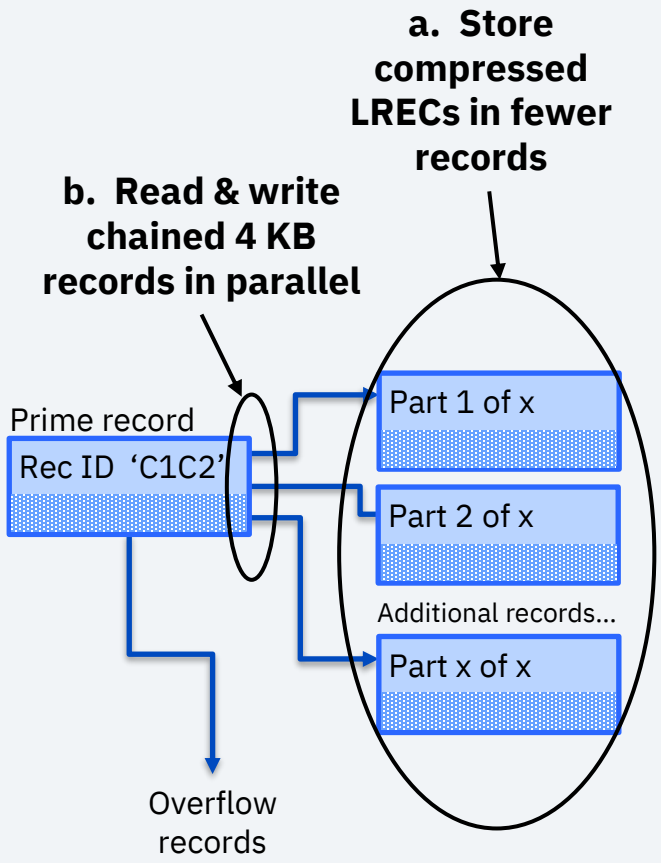
Note: An LREC address (SW00REC) would point into a z/TPFDF managed buffer instead of a core block

Writing a Compressed LREC Set



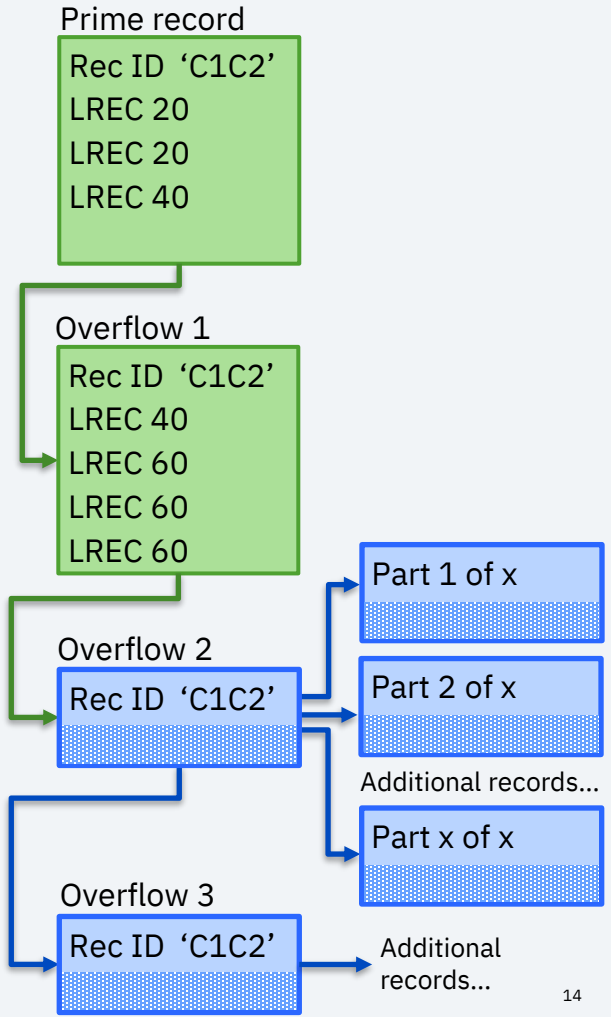
Reduced Response Times

- Reduce accumulated response time through a combination of compression and parallel I/O
  - **a. Fewer records:** Compress LRECs and store in fewer records than a standard subfile
  - **b. Parallel I/O:** Read and write multiple records at the same time instead of serially
- IBM z15 or later will be required so compression and decompression can be performed in hardware
- As long as applications use only z/TPFDF APIs to access and update LRECs, applications should not require any changes



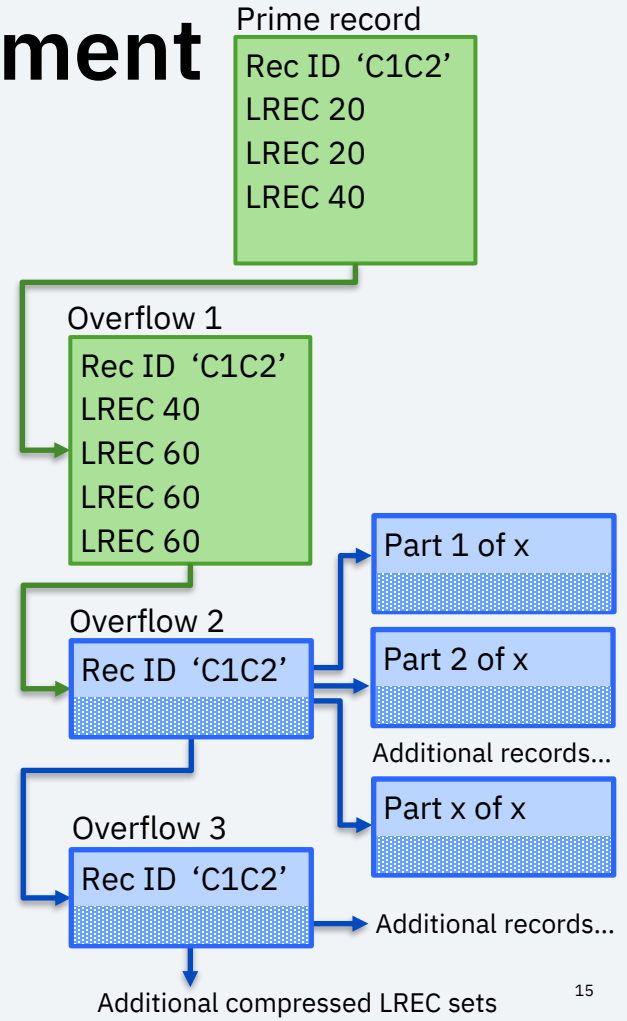
Some Uncompressed Data

- Some applications might read only the first few records in a subfile or only update LRECs at the beginning of the subfile
  - In this scenario, reading & writing records for a compressed LREC set could increase I/O
  - Option to keep the first N records uncompressed
  - For example, leave the prime and 1st overflow records uncompressed
- Periodically compress into a new or existing set as the amount of uncompressed data grows
  - Similar to z/TPFDF automatic pack operations to reduce unused space in a subfile



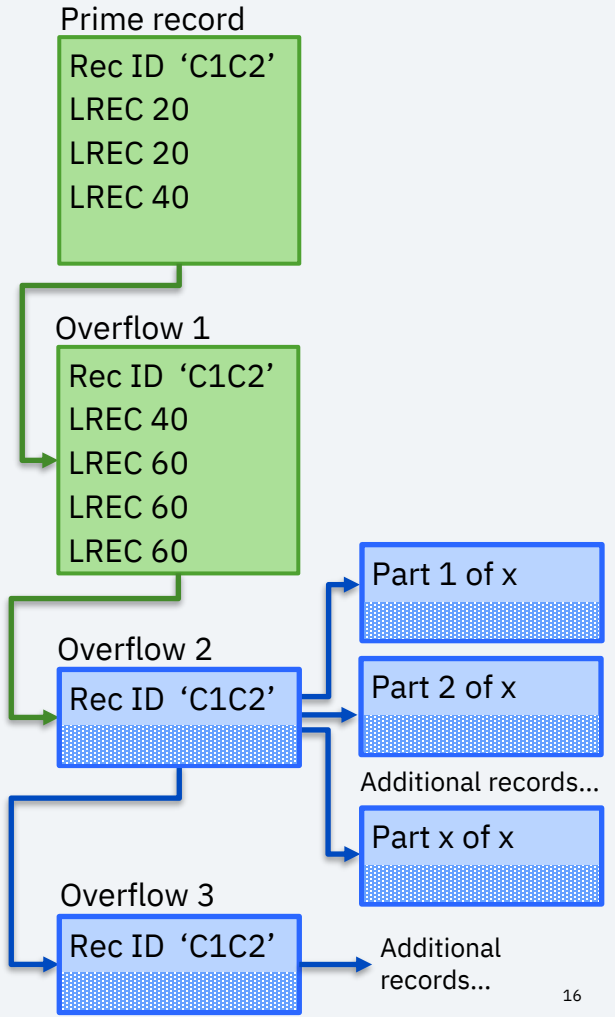
Example Service Time Improvement

- An uncompressed, standard 4 MB subfile can be read from DASD in 250 ms
  - 4 MB is 1000 records & service time is 0.250 ms
  - 1000 * 0.250 ms = 250 ms
- With the new subfile structure, a 4 MB subfile could be read from DASD in 9 ms
  - **96% reduction in accumulated service time**
  - Assumptions
    - With 80% compression, the subfile is stored in 2 uncompressed records and 10 compressed LREC sets, each with 20 chained 4 KB records
    - Pre-read the overflow record for the next set
    - Sufficient DASD modules to avoid abnormal queues

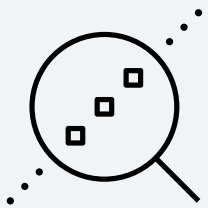


Current Assumptions

- Enabled and controlled through DBDEF parameters
  - Options to specify number of uncompressed records and other settings
- Migrate existing databases by using CRUISE PACK or similar functions
- Supported only for real-time (R-type) z/TPFDF files that do not use B+Trees
  - z/TPFDF encryption will be supported
- Goal is to not require application changes or database downtime

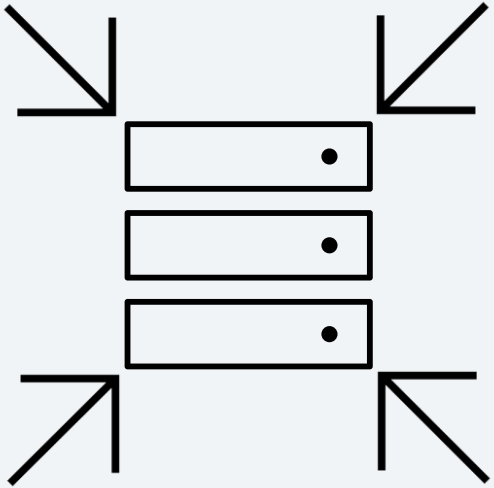


Access Patterns



- Access patterns are critical to understanding what design choices and configuration options are required to provide the right solution to meet your needs
- We need sponsor users to understand your applications' dominant access patterns
  - Read-only or update?
  - Types of updates (add to front, add to end, random)?
  - Do applications read most overflow records in a subfile or just a small subset?

z/TPFDF Storage Improvements



As-Is User Story

Calvin notices that the number of available pool records is decreasing faster than expected. Calvin doesn't have a budget to add more DASD storage and a DASD refresh is still years away.

Anna and Andres tell Calvin that business growth and regulatory requirements have increased the number of subfiles in their database, causing their databases to grow faster than expected.

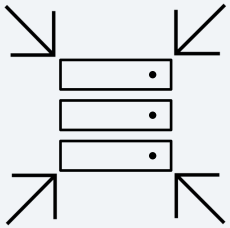
Calvin and Andres need to figure out how to add more physical storage and make it available to the system before available pools fall below critical levels.

Calvin
Capacity planner

Anna
Application architect

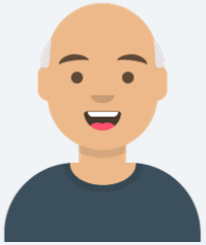
Andres
Database admin

Value Statement



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.**

To-Be User Story



Calvin
Capacity planner

Anna
Application architect

Andres
Database admin

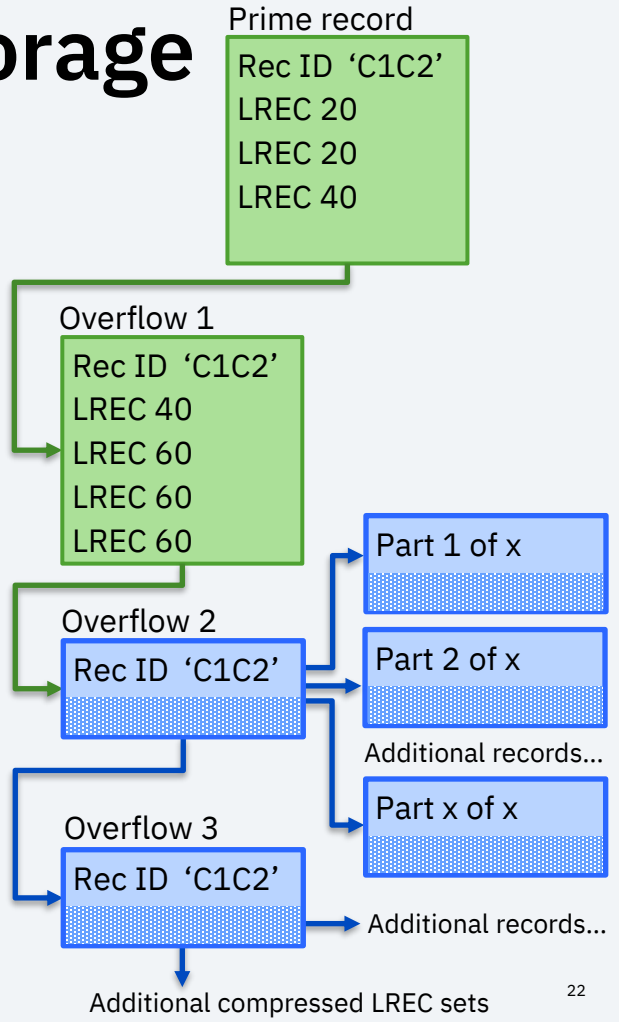
The number of available pool records is decreasing faster than expected and Calvin doesn't have a budget to add more storage or refresh DASD.

Anna informs Calvin that business growth and regulatory requirements are causing the number of subfiles to grow faster than expected.

Andres informs them that migrating their databases to use a new z/TPFDF subfile structure would let them store the same amount of data in less space. The database could continue growing **without requiring application changes, database downtime, or more DASD storage.**

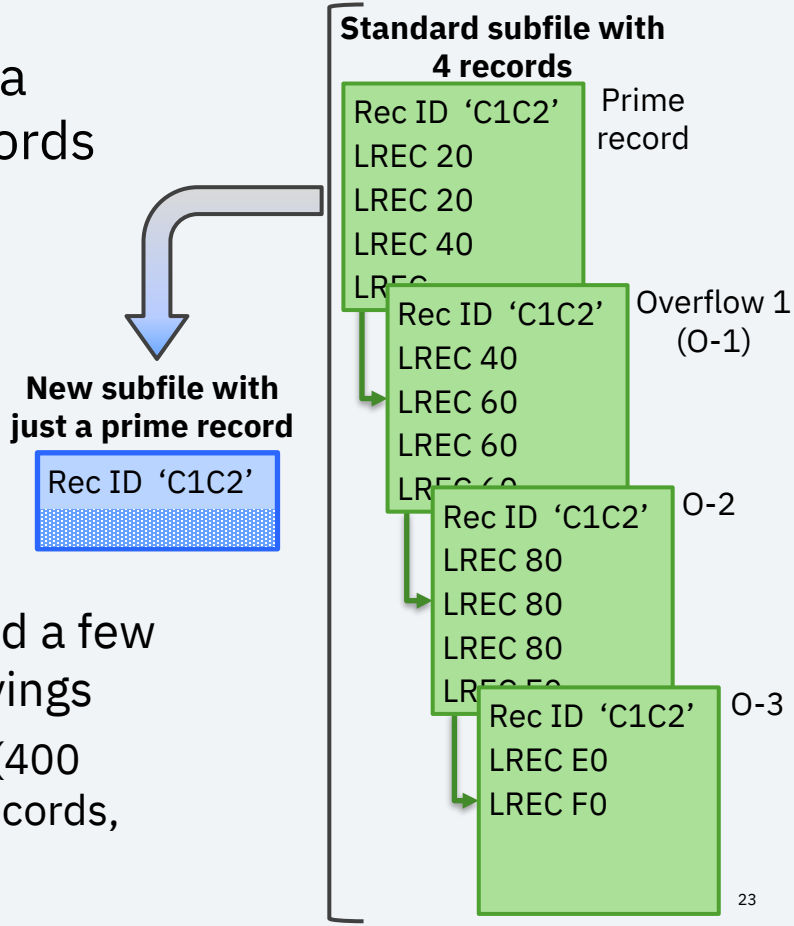
Example Reduction in DASD Storage

- An uncompressed, standard 4 MB subfile uses 1000 records
- With the new subfile structure, a 4 MB subfile could be stored using 212 records
  - **78% reduction in number of records**
  - Assumption: With 80% compression, the subfile is stored in 2 uncompressed records and 10 compressed LREC sets, each with 20 chained 4 KB records
- Reduced storage needs for **every copy** of the subfile
  - Prime and duplicate modules, VFA and DASD control unit cache, disaster recovery, point-in-time copies like Flashcopy and Safeguarded Copy, and test systems

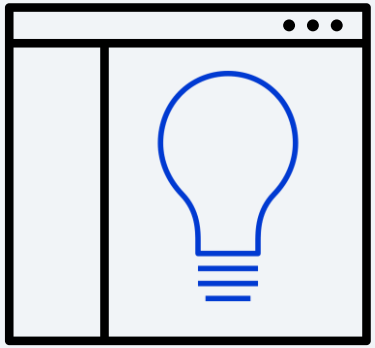


Savings for Subfiles with a Few Overflows

- A standard subfile with 14 KB of LREC data could be stored in as few as four 4 KB records
- With the new subfile structure 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 100's of 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



Smarter Applications

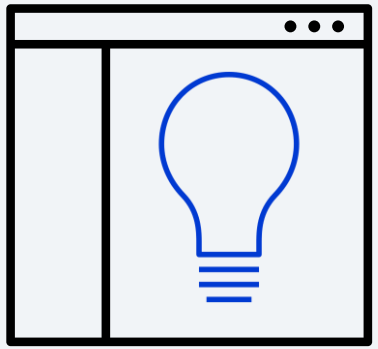


Value Statement

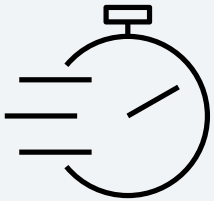
Smarter applications can make better business decisions by **accessing significantly more data with no additional latency.**

Smarter Applications

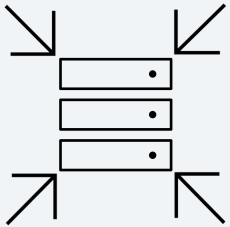
- With the ability to store more data and quickly access that data, what does that enable?
  - Optimize decision making for your business by accessing more data per transaction
  - Improve your customers' experience by returning more information or providing more context
  - Save more information per transaction to use as input into AI and analytics



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 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

Begins

3Q 2024

Interested? Contact

Chris Filachek (filachek@us.ibm.com)

Thank you