z/TPFDF Update

2022 TPF Users Group Conference
March 27-30, Dallas, TX
Database/TPFDF Subcommittee

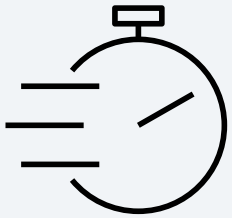
Chris Filachek

IBM Z

IBM

z/TPFDF Recoup Optimized Chain Chasing

Problem Statement



Typically, recoup has a time window when it can run. If recoup exceeds the time window, other work is impacted.

As databases grow larger, it becomes difficult to keep recoup runtime within the allowed time window.

Pain Points

Andres
database admin

I am estimating that one of our z/TPFDF databases will grow to 100’s of billions of records in the next few years. Recoup might take over a day to read just 10’s of billions of records in our environment.

Our recoup runtime window is 10 hours and longer recoup runtimes will impact other utilities and business-critical processing.

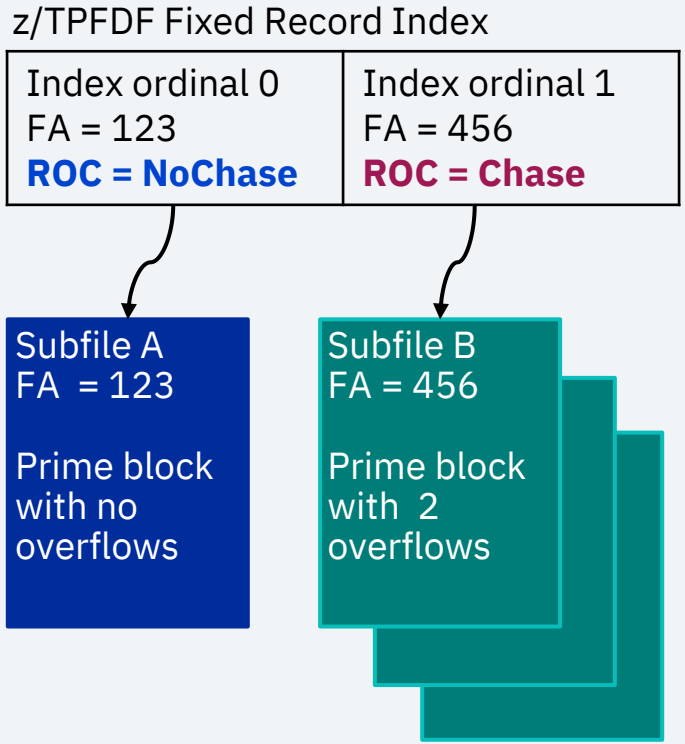
Value Statement

A database administrator can enable Recoup Optimized Chain Chasing for an indexed z/TPFDF database, allowing recoup to chain chase billions of single-record detail subfiles in a fraction of the time compared to existing recoup processing.

Assuming a database with 1B index ordinals, 80B single-record detail subfiles, and recoup reading at 500K IOPS …

- ✗ Recoup chain chase takes almost **2 days** with standard chain chase
- ✓ Recoup chain chase takes about **30 minutes** with optimized chain chase

How it works



Recoup optimized chain chase (ROC) conditionally reads records in the database

- ROC Flag indicates if recoup needs to read the detail subfile referenced by the index LREC
  - **NoChase** – Only 1 record. Mark pool as in-use and skip read
  - **Chase** – Multiple records (overflows or LLRs). Read all records in chain

z/TPFDF stores a technical LREC (TLREC) in each detail subfile with index information

- z/TPFDF uses the TLREC to find the index LREC and update the ROC Flag as needed

Enable recoup optimized chain chase in z/TPFDF

- No application changes are required
  - Supported with most z/TPFDF APIs
- Use on both existing and new z/TPFDF single-level indexed databases
  - Enabled through DBDEF parameters on the index file, ID1=(RCPOPTIDX), and the detail file, IMI=#BIT4
  - Index files must used 8-byte forward index path references (FAL=8)
  - Detail files must be R-type files with variable length LRECs and allow TLRECs up to ID X'15'

See z/TPFDF recoup optimized chain chasing for more information on how to plan for, enable, and use this support

Enable optimized chain chase in recoup

- Use the ZRECP PROFILE command and DFOPTMZ parameter to enable optimized chain chasing in recoup
  - Enable optimized chain chase in the DBDEFs, but wait until you are ready to use in recoup
  - Optimized chain chase used only for single-record subfiles indexed after this option is enabled in the DBDEFs
  - Standard chain chase used for all other subfiles
- Recoup will not check for broken chains in subfiles that use the optimized chain chase
  - CRUISE VERIFY can be run as a low priority utility in the background to check for broken chains in all subfiles

ZUDFM ACCESS Pain Points

Andres
database admin

I can’t use ZUDFM ACCESS to index an existing subfile. ZUDFM ACCESS can index a subfile only when the subfile is being created at the same time.

I can use ZUDFM ACCESS to deindex a subfile by using the RELEASE parameter. However, it does NOT delete the subfile or return pools, which is what “RELEASE” might imply.

ZUDFM ACCESS Enhancements - Indexing

ZUDFM ACCESS command supports a new INDEX parameter

- Use the INDEX parameter to index an existing subfile
- First access a subfile and then index it on the desired path

```
==> ZUDFM ACCESS FD01/ALG-42

UDFM0301I          FILE INFORMATION DISPLAY
FILE IDENTIFIER                 FD01
. . .


==> ZUDFM ACCESS FD01/INDEX/PATH1/ALG-|XYZ|01

UDFM0330I 11.33.32 THE SUBFILE WAS SUCCESSFULLY INDEXED ON PATH 1 BY USING
 ALGORITHM |XYZ|01
```

ZUDFM ACCESS Enhancements - Deindexing

ZUDFM ACCESS RELEASE parameter renamed to DEINDEX

- Use the DEINDEX parameter to deindex an existing subfile
- First access a subfile and then deindex it on the desired path

```
==> ZUDFM ACCESS FD01/ALG-42

UDFM0301I          FILE INFORMATION DISPLAY
FILE IDENTIFIER                 FD01
. . .


==> ZUDFM ACCESS FD01/DEINDEX/ALG-42

UDFM0330I 11.33.32 THE SUBFILE WAS SUCCESSFULLY DEINDEXED ON PATH 0 BY USING
 ALGORITHM 42
```

Summary

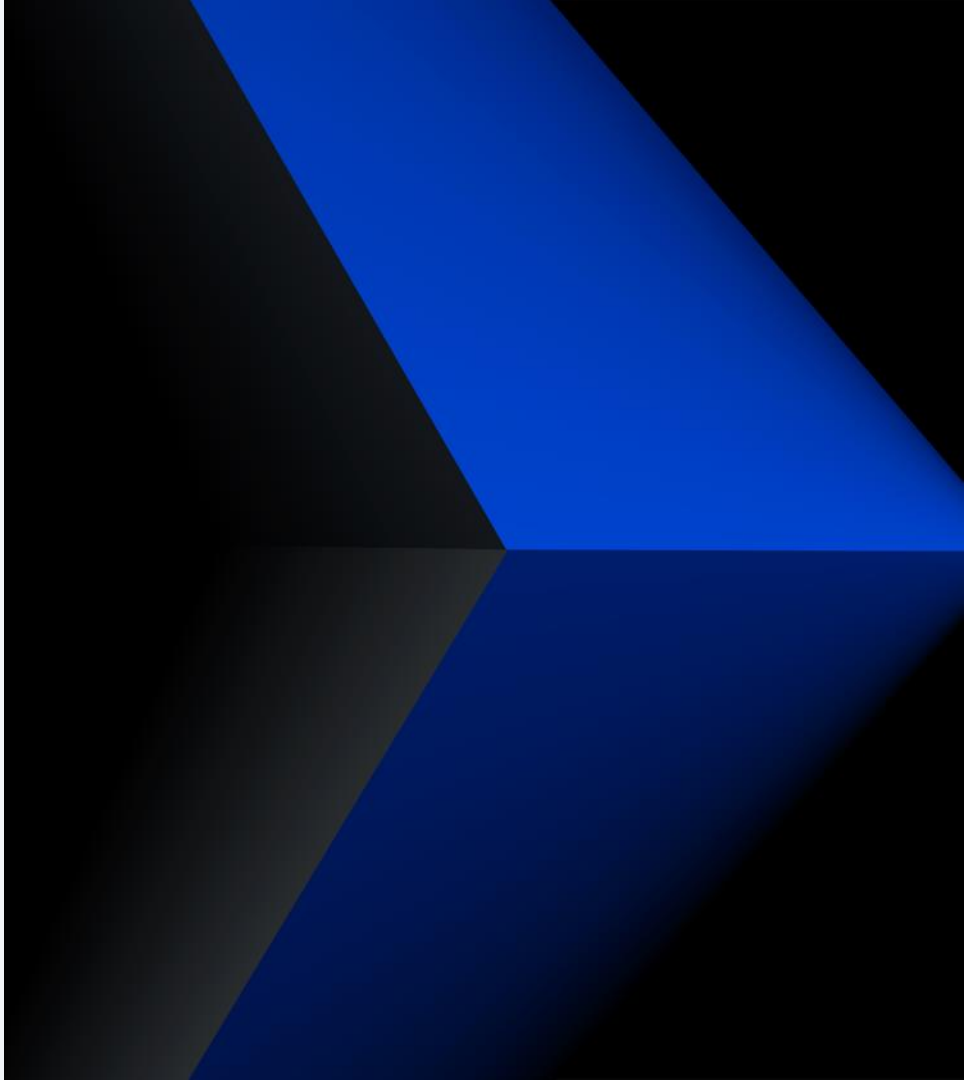
- New recoup optimized chain chase option, which allows recoup to chain chase single-record subfiles in a fraction of the time

- New and updated index and deindex parameters for the ZUDFM ACCESS command

APARs PH30068 and PJ46413 (November 2021)

Disclaimer

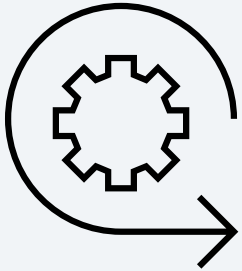
Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.



ZUDFM INIT Enhancements

Subsystem Users and Cancel Options

Problem Statement



ZUDFM INIT initializes the z/TPFDF file only for the subsystem user where the command is entered.

As Is User Story

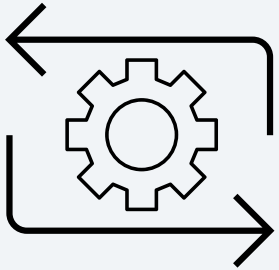
Andres
database admin

I need to initialize a subsystem user unique z/TPFDF file for all subsystem users (SSUs). With dozens of SSUs, I need to enter dozens of commands to fully initialize this file.

Also, one of the SSUs is dormant and ZUDFM INIT can't initialize records on a dormant SSU.

I can't use the ZIFIL command because it doesn't correctly initialize records for use by z/TPFDF. For example, it doesn't setup the records for z/TPFDF 8–byte headers (FARF6) or z/TPFDF encryption.

Value Statement



An operator can enter a single ZUDFM INIT command to initialize a z/TPFDF file for all subsystem users on the current subsystem or initialize a z/TPFDF file for a dormant subsystem user.

To Be User Story

Andres
database admin

I need to initialize a subsystem user unique z/TPFDF file for all subsystem users (SSUs). I can enter just one ZUDFM INIT command to initialize all records for all SSUs.

When ZUDFM INIT initializes records for all SSUs, the records for dormant SSUs are initialized as well. If I need to initialize records for just the dormant SSU, ZUDFM INIT allows me to do that as well.

...and since you're already updating ZUDFM INIT

As Is

I have to enter ZUDFM INIT twice to start the initialization, but there's no clear way to cancel the first ZUDFM INIT. The first ZUDFM INIT times out in 2 minutes, but the request is still in the database and interferes with future ZUDFM INIT commands.

To Be

I can use a new parameter with ZUDFM INIT to cancel the first ZUDFM INIT command. If the first command times out after 2 minutes, the request is automatically removed from the database so future ZUDFM INIT commands are not impacted.

We want sponsor users!

Our development cycle is driven by your feedback.

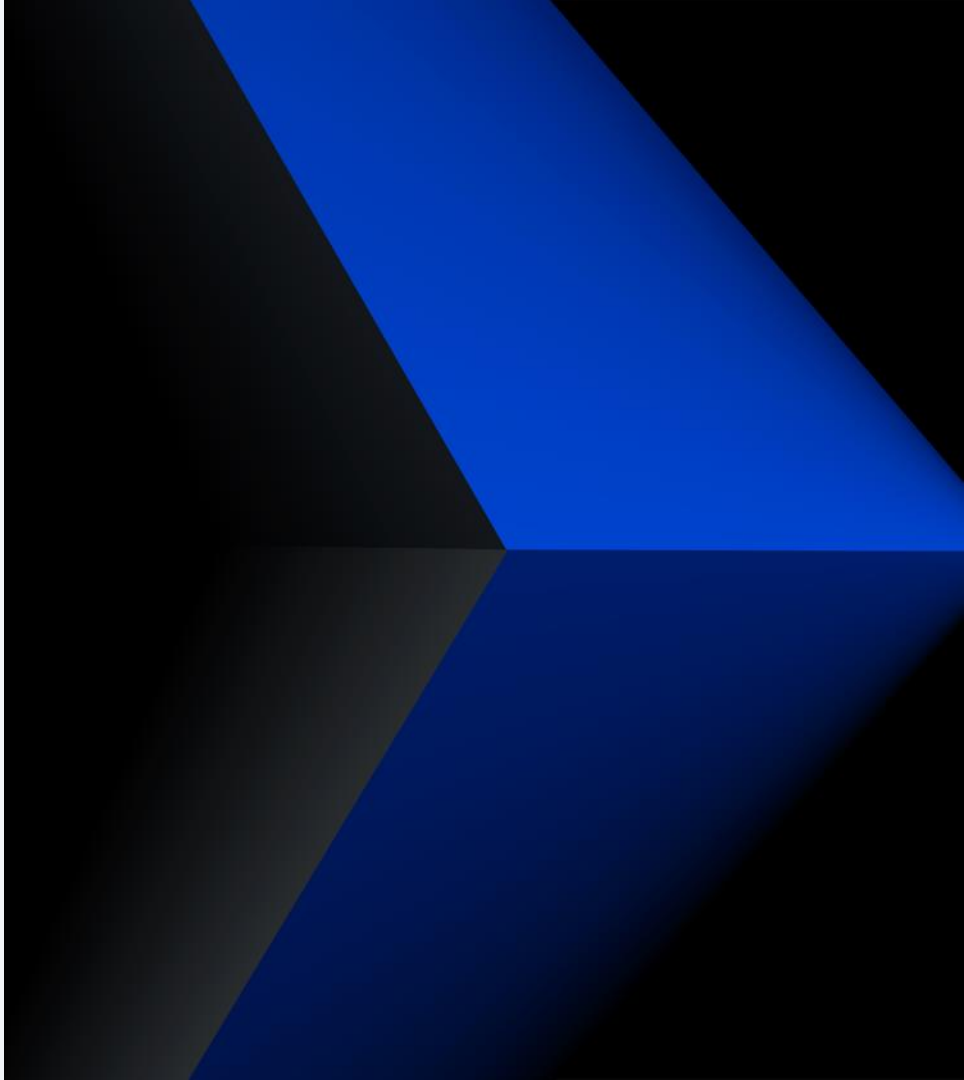
We are looking for sponsor users to assist in design and implementation, targeting the following personas:

- Database administrators
- Operators

We expect to begin engaging with the sponsor users in 2Q 2022.

If you are interested in participating as a sponsor user, please contact:

Chris Filachek (filachek@us.ibm.com)

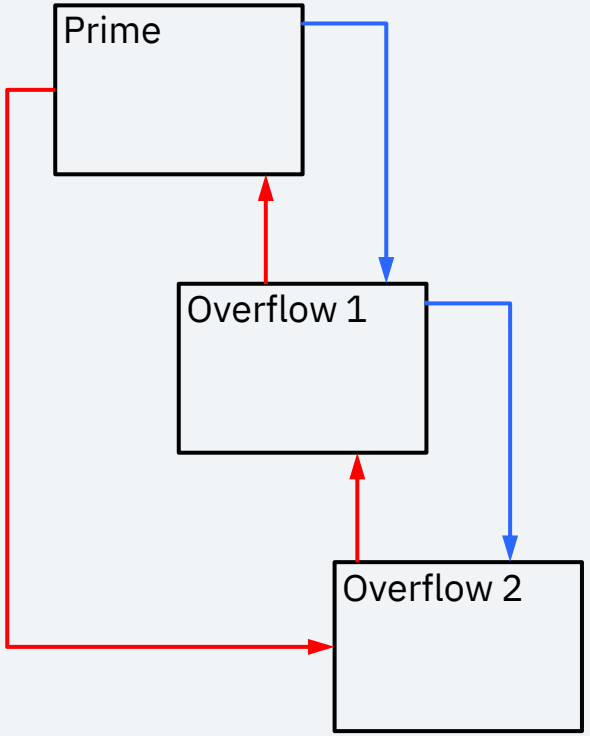


CRUISE Enhancements

Validating Backward Chains

Problem Statement

The z/TPFDF CRUISE utility does not use or validate backward chains in z/TPFDF subfiles. CRUISE only follows forward chains.



As Is User Story

Andres
database admin

Due to a recent issue, some of the backward chains in our z/TPFDF subfiles were corrupted.

We were alerted to the corruption because z/TPFDF issued an error when an application asked z/TPFDF to read backwards.

Packing a subfile appears to quietly fix the issue, but we don’t have a way to determine the extent of the corruption.

Value Statement

The z/TPFDF CRUISE utility (VERIFY, PACK and CAPTURE functions) will issue an error message if an incorrect backward chain is found in a z/TPFDF subfile

- Any pack of the subfile will fix incorrect backward chains
- Additional error messages will be issued if the IBM reserved STDAUT bits are in use

To Be User Story

Andres
database admin

Due to a recent issue, some of the backward chains in our z/TPFDF subfiles were corrupted.

We were alerted to the corruption because z/TPFDF issued an error when an application asked z/TPFDF to read backwards.

We used CRUISE VERIFY to identify the affected subfiles and specific blocks in the chain. We then used z/TPFDF commands to pack only those subfiles to fix the corruption.

We want sponsor users!

Our development cycle is driven by your feedback.

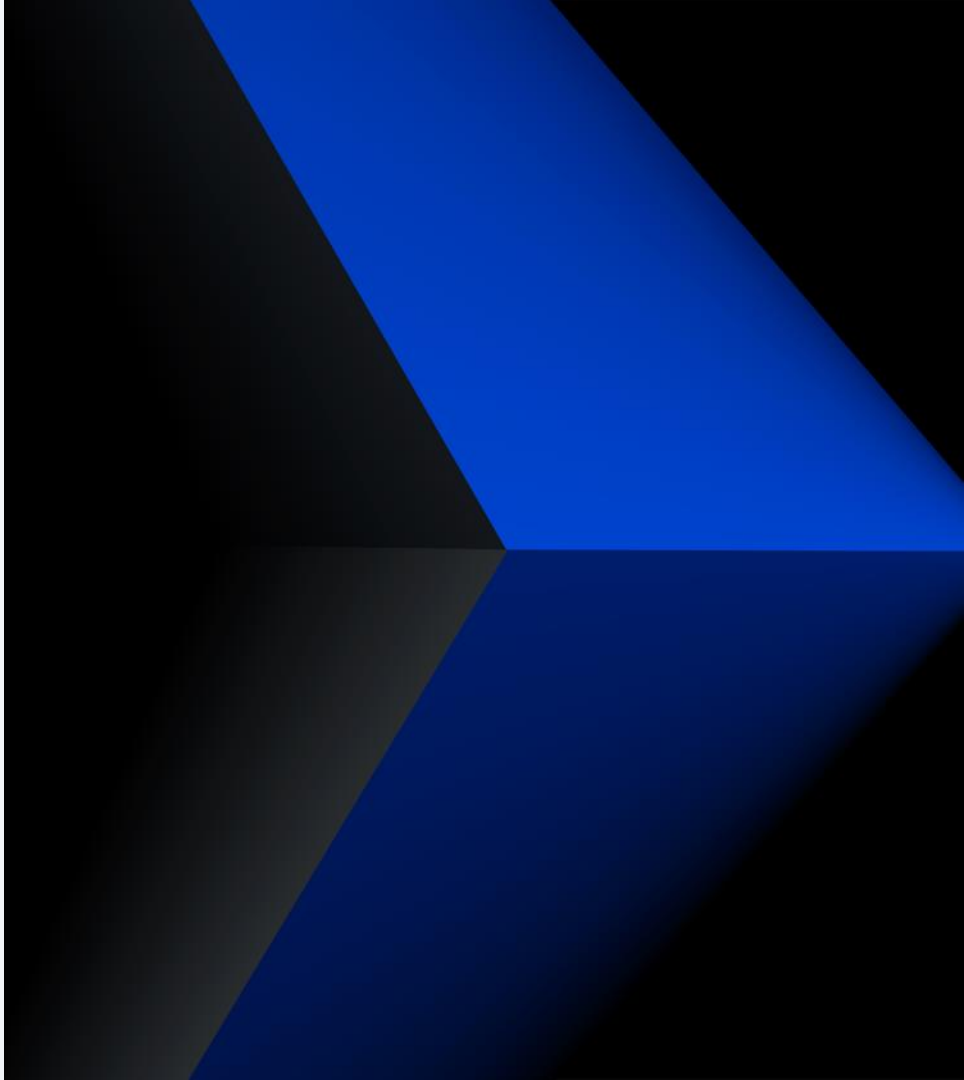
We are looking for sponsor users to assist in design and implementation, targeting the following personas:

- Database administrators
- Coverage programmers

We expect to begin engaging with the sponsor users in 2Q 2022.

If you are interested in participating as a sponsor user, please contact:

Chris Filachek (filachek@us.ibm.com)



Thank you