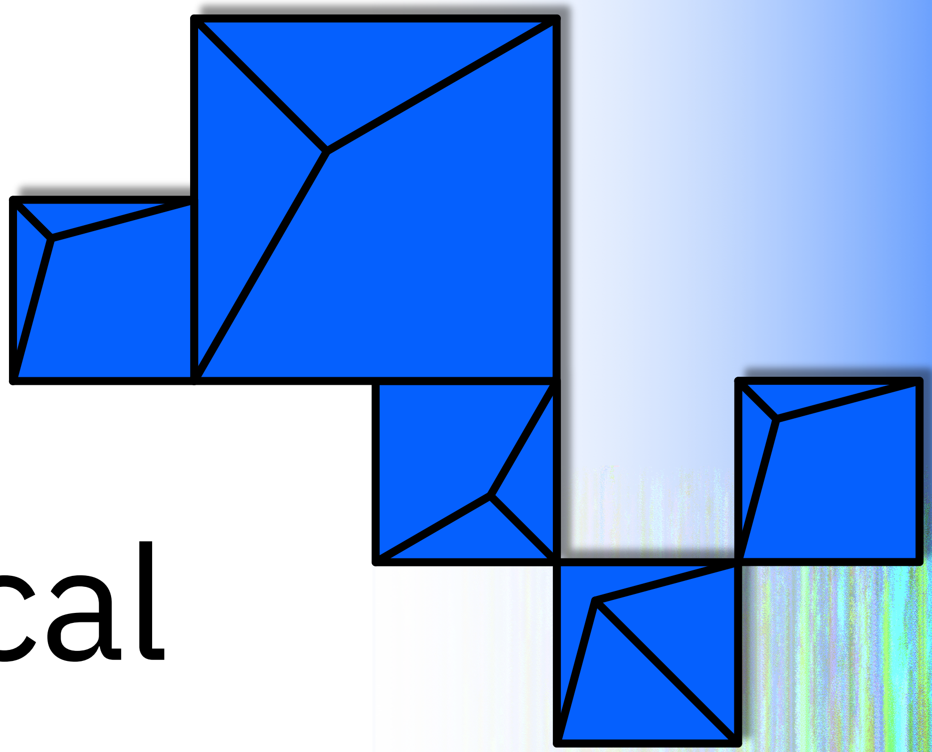
# Recoverable Logical Record Cache



Chris Filachek

# Agenda

- Cached data, IPLs, and Pain Points
- Preserving cached data across IPLs
- Defining a logical record cache
- Managing caches with definitions
- Migrate your logical record cache
- Up Next: Changing cache sizes

## Problem Statement

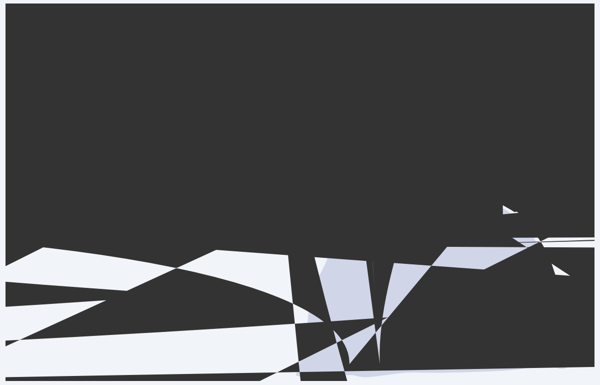
Logical record caches and cache entries are lost across an IPL.

- Applications can’t make effective use of an empty cache.
- This can result in high resource utilization (CPU, DASD, etc.) until the cache becomes sufficiently populated.
- For large caches, it can take a significant amount of time to naturally repopulate the cache.

## Operators

- Because caches are lost across an IPL, we have to monitor system resources very closely after every IPL and watch for high resource utilization and input list shutdown conditions.

- If resources are low, we have to quickly adjust resources (add CPU), restrict traffic, stop or delay utilities, etc.



Derrick

**Operator**

## Coverage Programmers

- Restricting traffic can result in service disruptions to our customers.
- Some of our logical record caches are GB in size, so the impact (extra resources and service disruptions) could last several hours after an IPL.

Carol

**Coverage Programmer**

## Capacity Planner

- To help mitigate the impact of an empty cache after an IPL, we have to activate extra resources like CPU.

- These extra resources are only needed immediately following an IPL and need to be deactivated after the cache is sufficiently populated.

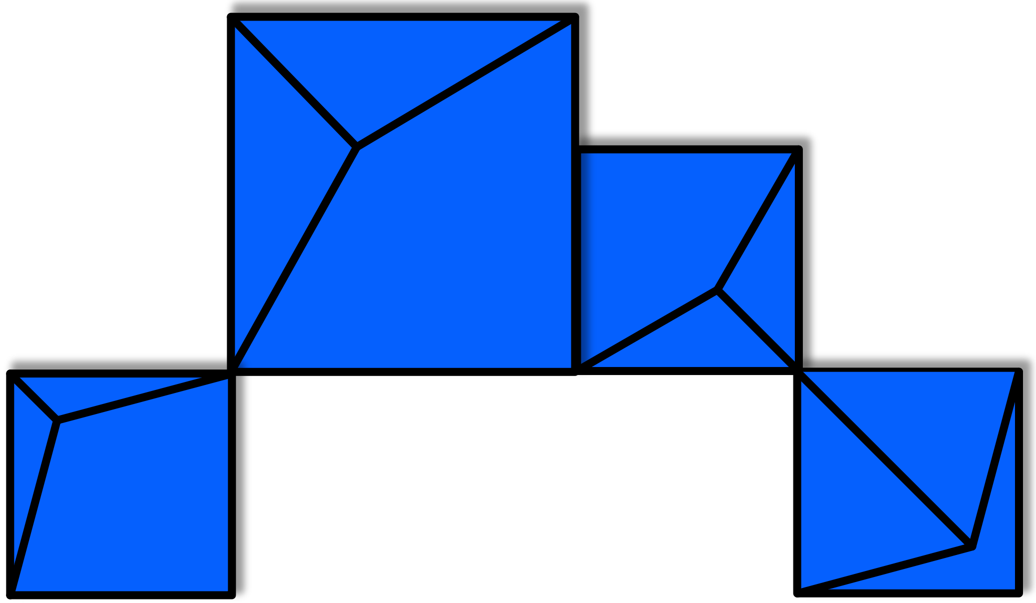
Calvin

**Capacity Planner**

## Value Statement

Allow the contents of logical record cache to be preserved across most IPLs so the cache is effective as soon as z/TPF reaches NORM state.

- Allow z/TPF applications that use logical record cache to immediately recover to steady state following most unplanned and some planned IPLs.
- Preserve cached data across IPLs so the extra monitoring, resource usage, and service disruptions due to empty caches can be avoided.



# Preserving cached data across IPLs

## Before you begin…

- Install support for recoverable system heap on your z/TPF system
  - APAR PJ45598
  - z/TPF Level 2019 (November 2019)
- Allocate sufficient recoverable system heap for your anticipated needs
  - See "Using recoverable system heap" in the z/TPF Knowledge Center for more information.

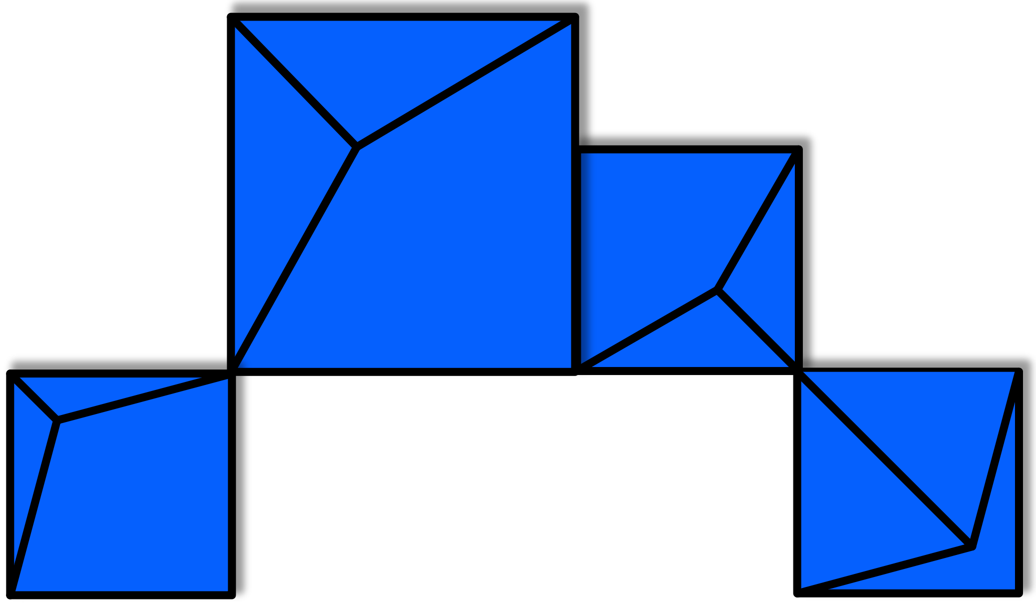
## Use recoverable logical record caches

- Specify the recoverable system heap option when creating the cache
  - Cache is allocated using recoverable system heap instead of 64-bit (non-recoverable) system heap

- After an IPL, allocating the cache automatically looks for existing cache data in recoverable system heap and reattaches if found
  - Applications use the same cached data that existed prior to the IPL
  - Always get a new cache token after an IPL
    - Cache tokens change across IPLs

## Reduce Monitoring

- After most unplanned and some planned IPLs, applications using recoverable logical record cache quickly reach a steady state
  - Cached data is preserved across most IPLs and the cache immediately provides data to applications
  - No extra monitoring or service disruptions due to ineffective caches



## Defining a logical record cache

## Background

Logical record cache attributes are set by the new cache functions in application code.

- Cache name and type
- Cache size
- Entry count and size
- Cast out time

## Application Architects

- Because logical record cache attributes are set in application code, we have to request application changes to update most cache attributes.

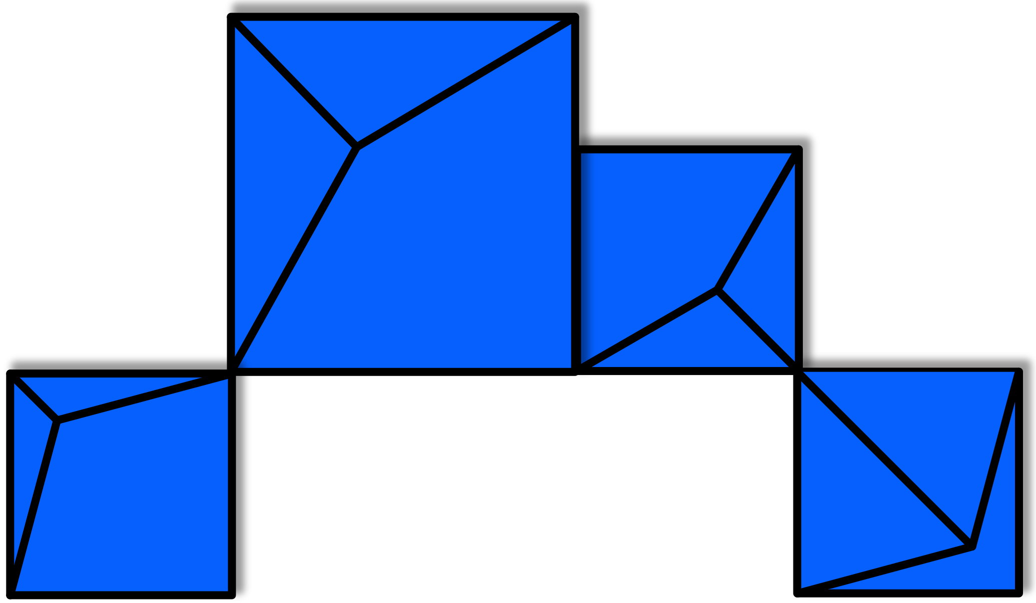
- Changing even a single attribute like the cast out time forces us to go through application code, build, and test cycles, which makes it time consuming to try different attributes or make production changes.

Anna

**Application Architect**

## Value Statement

Allow caches to be defined and managed through configurations that can be easily changed without requiring application changes.



## Managing caches with definitions

## Cache Definitions

- Managed through the ZCACH command
- Defines the attributes of a logical record cache
  - Same cache attributes that are set with the new cache functions
    - Cache name, type, and size
    - Entry count, entry size, and key lengths
    - Castout time and program
    - Memory type: Recoverable or 64-bit (non-recoverable) system heap
- Attributes no longer have to be embedded in applications
  - Avoid application changes when updating cache attributes

## Creating a cache definition: ZCACH DEFINITION

- Create a cache definition with a cache name and type
  - Cache type is processor unique or shared
  - Creating a definition does not allocate the cache in memory
- Set the attributes for the logical record cache
  - Cache size, entry count, entry size, and key lengths
  - Castout time and program
  - Memory type: Recoverable or 64-bit (non-recoverable) system heap
- Example: Create the definition for MYCACHE and use recoverable system heap

```
ZCACH DEFINITION CREATE MYCACHE CACHETYPE-UNIQUE
ZCACH DEFINITION ALTER MYCACHE MEMTYPE-RECOVERABLE
```

## Creating and Deleting the cache in memory

- Create a cache in memory: ZCACH CREATE
  - Uses the cache definition to create the logical record cache in memory
  - Automatically recreates the cache after every IPL
    - If the cache was created using recoverable system heap, find and reattach to the recoverable system heap
  - Example: Create the MYCACHE cache

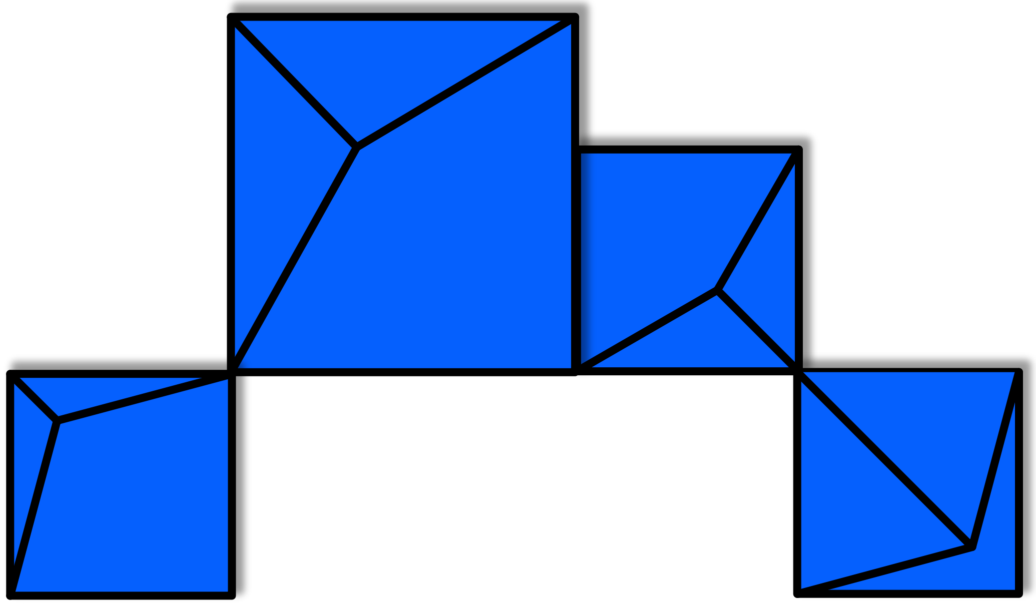
```
ZCACH CREATE MYCACHE
```

- Delete a cache from memory: ZCACH DELETE
  - Deletes the logical record cache from memory
  - Stops system restart from recreating the cache following an IPL
  - Does not delete the cache definition

## Application Considerations

- For caches created by ZCACH, applications must use the tpf_cacheNameToToken function to get the cache token
  - Cache token is needed to call other cache functions (read, update, etc.)
  - The cache functions, newCache() and tpf_newCache_ext(), return cache tokens only for caches that were created by those functions

- For more information on using the cache definitions, see "Manage logical record caches and logical record cache definitions" in the z/TPF Knowledge Center



## Migrate your logical record cache

## Migrate to Definitions and Recoverable System Heap

1. Update applications to get the cache token using the tpf_cacheNameToToken function
   - No application changes are required to read, update, delete or flush cache entries. Cache entry functions work the same for:
     - Recoverable and non-recoverable caches
     - Caches created by ZCACH, newCache(), and tpf_newCache_ext()
2. If this is an existing cache, use the ZCACH command with the DELETE parameter to delete the cache from memory
   - The cache is not available to applications after this step and is created as an empty cache in the next step
3. Create a cache definition and set all required attributes using the ZCACH command with the DEFINITION parameter
   - Specify MEMTYPE-RECOVERABLE with the ZCACH command to create the cache using recoverable system heap
4. Use the ZCACH command with the CREATE parameter to create the cache in memory using the definition

## If you need to continue using the new cache functions…

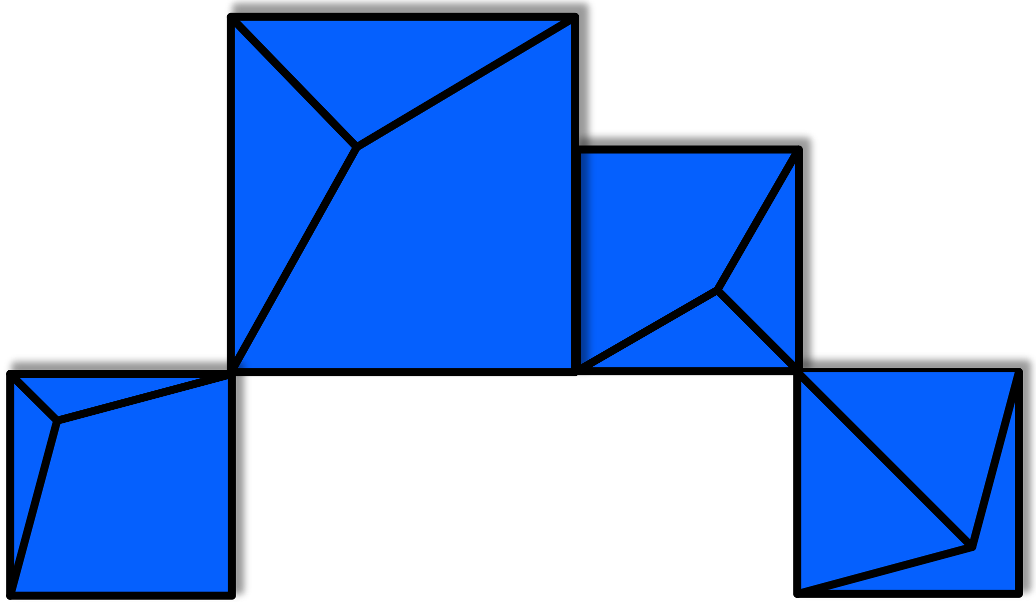
- Specify the CACHE_USE_RECOVERABLE_SYSTEM_HEAP option on the newCache() and tpf_newCache_ext() APIs
  - Allocates the cache using recoverable system heap
  - An error is returned if
    - Cache is already allocated using 64-bit system heap (non-recoverable)
    - The cache size is larger than the available recoverable system heap
  - See the extended option parameter, cacheExt, for newCache() and tpf_newCache_ext() APIs

## Recoverable Logical Record Cache (RLRC) support

- APAR PJ45782
- z/TPF Level 2019 (December 2019)

## 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.



## Up Next: Changing the cache size

## Problem Statement

- Existing caches can lose effectiveness as database use expands but cache size stays the same
  - Cache size becomes too small and valuable cache entries may be forced out of cache to make room for new entries
- To change the size of a logical record cache, the cache must be deleted and recreated.
  - Although planned, this still results in an empty, ineffective cache
  - Might require a significant amount of time and resources to repopulate the cache

## Pain Points

- Because the cached data is lost, changing the cache size might require as much planning, time, and resources as a planned outage
- May be easier to delay cache size changes to the next planned outage and “live with” the performance penalty of an undersized cache for an extended period of time

## Value Statement

Easily adjust to workload growth and improve cache effectiveness by increasing the cache size without losing cached data.

## Technical Details

Current approach allows the cache to be resized up to maximum number of entries

- Cache size is based on entry size, number of entries, and internal cache structures
- A new customer defined attribute sets the maximum number of entries for a cache
  - Used to allocate internal structures
- Existing number of entries attribute used to allocate cache entries in memory
- Add cache entries up to the maximum entries attribute without deleting the cache or losing cached data

## Example scenario:

1. Using the ZCACH command and cache definitions, create a cache with the maximum number of entries larger than number of entries
   - Number of entries: 1 million
   - Maximum number of entries: 2 million
2. Over time, workload growth reduces effectiveness of the cache
3. Using the ZCACH command, increase the number of entries in the cache definition
   - Number of entries: 1.5 million
4. Using the ZCACH command, apply the updated number of entries to the cache in memory
   - Adds 500,000 new entries to the existing cache
   - Does not affect existing entries

## Technical Details

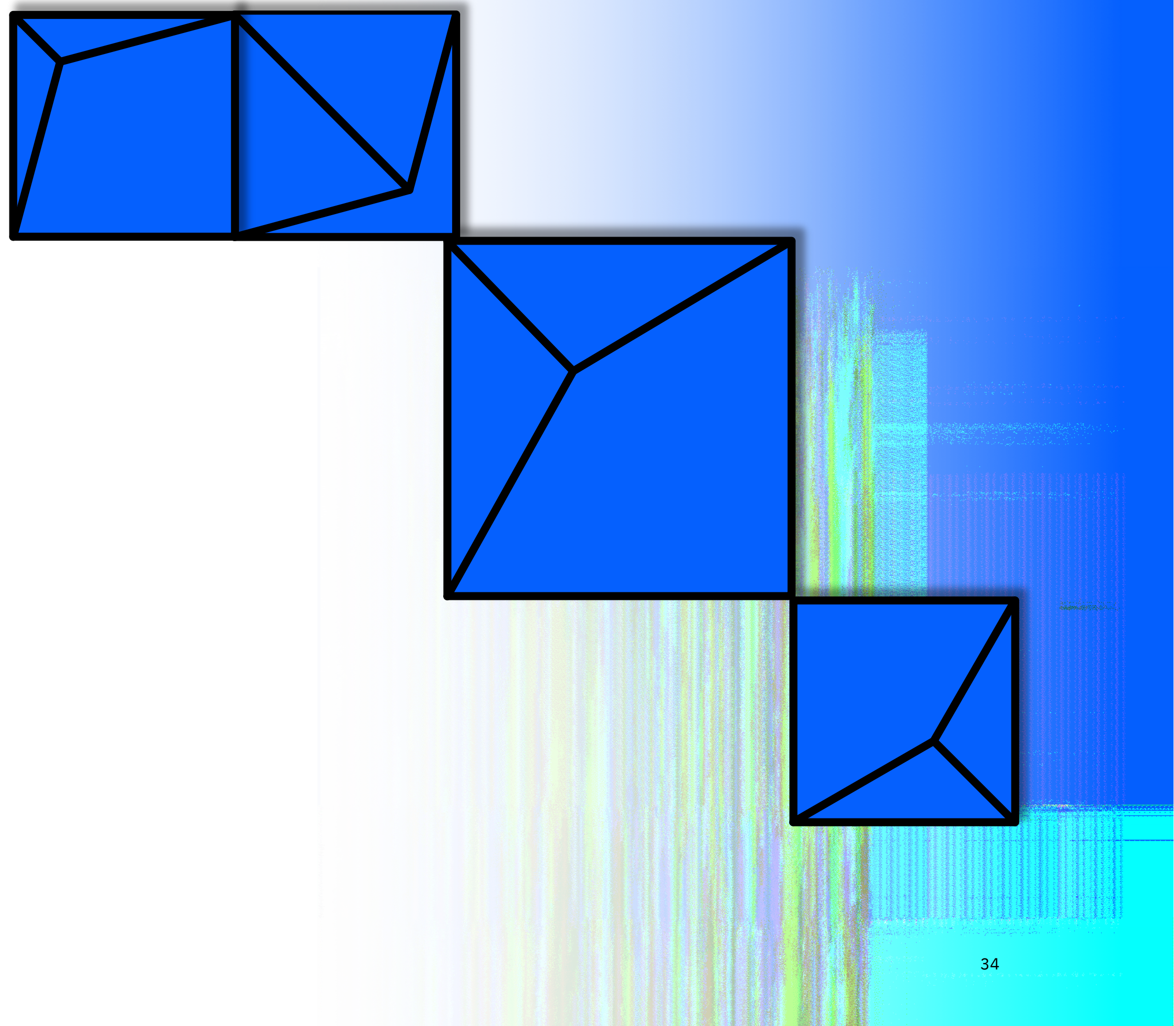
- Plan for future growth by setting maximum entries larger than number of entries
- Resizable caches can use 64-bit system heap or recoverable system heap
- Resizable caches must be
  - Created from a cache definition
  - Processor unique
  - A traditional logical record cache (entry size less than 4K)

## What’s next?

- If you are interested in participating as a sponsor user for logical record cache, contact us at:
  - Chris Filachek: filachek@us.ibm.com
  - Danielle Tavella: Danielle.Tavella@ibm.com

# Thank You

Questions? Comments?



## Virtual TPFUG Q&A

Summary of Q&A from the virtual TPFUG event:

| Question | Answer |
|---|---|
| Q: However that Cache may be out-dated after IPL? | A: For caches that use 64-bit (non-recoverable) system heap, the cache is created empty after every IPL (no data in the cache).<br><br>For caches that use recoverable system heap, the data should not be out-dated if the cache data is recovered across an IPL. When the cache data is recovered, the recovered data includes the cast out times set when the cache entries were last created or updated. Some cache entries might reach or exceed their cast out times during the IPL and those entries would be invalid. Assuming the cast out times are not set to a very small time value, most cache entries should still be valid after an IPL.<br><br>If a cast out time is not set, then logical record cache assumes that the recovered cache entries are valid across an IPL and the application is responsible for managing out-dated entries by updating or deleting out-dated entries as needed.<br><br>Recoverable system heap is only supported with processor unique caches. It is not supported with processor shared caches, so cached data is not recovered and can't be out-dated after an IPL with a shared cache. |
| Q: does this use TE or GP MIPS? | A: Logical record cache uses GP MIPs. Even though logical record cache uses GP MIPs, caching is meant to make your z/TPF system more efficient by providing cached "answers" for your applications and avoid system resource usage that would normally be used by that processing.<br><br>When your application receives a request like an availability, processing the availability request consumes some amount of system resources (MIPs, DASD I/O, etc.). For some requests like availability and without any caching, your applications might consume MIPs, DASD I/O, or other system resources to compute the same answer hundreds or thousands of times a second.<br><br>If your applications use logical record cache to save frequently used "answers", your applications can compute the answers once, reuse those answers thousands of times, and save the system resources normally used by that processing. These "answers" can be complete responses or just a part of the processing for a request. |

# Trademarks