Dynamically Resizable Logical Record Cache

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

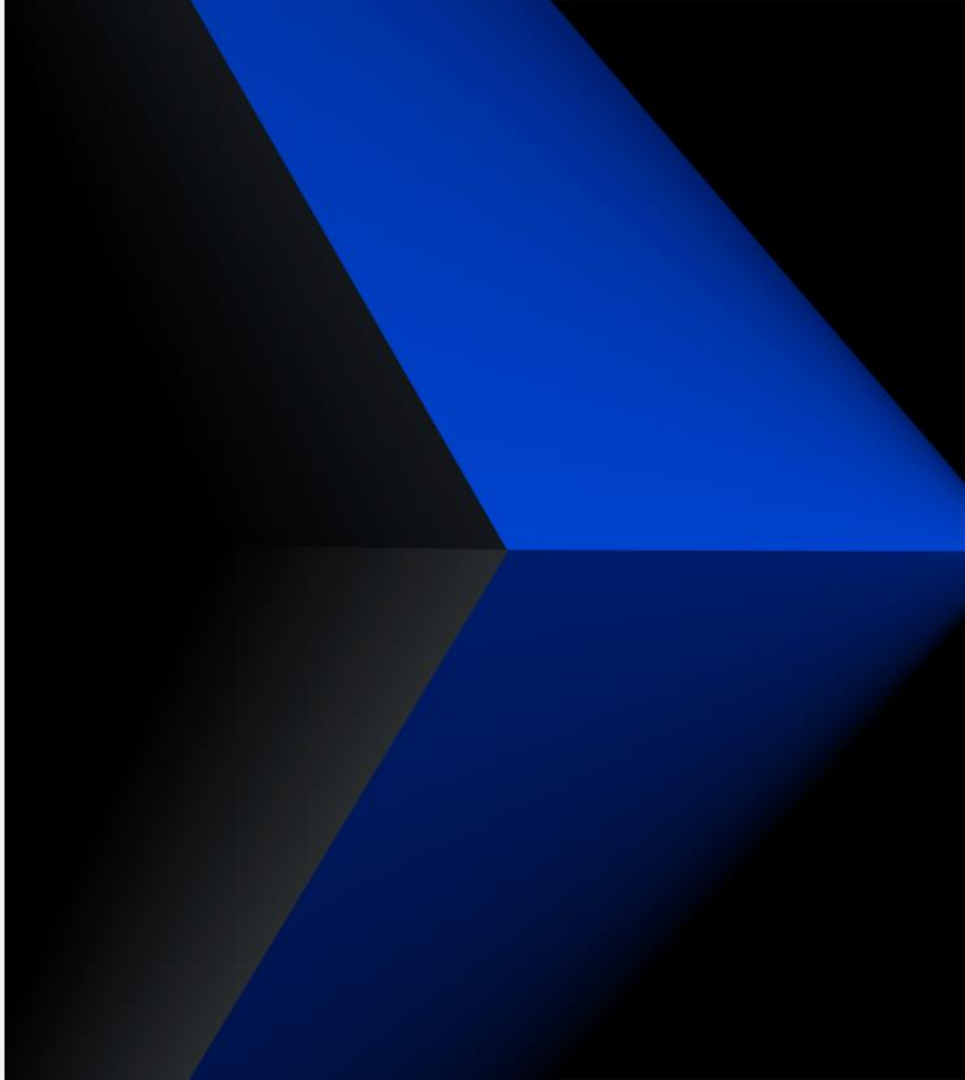
—

Claire Durant

IBM Z

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.



Logical record caches

- Use a logical record cache for high-speed access to data.
- The data in a cache is globally accessible to all programs.
- IBM system code uses logical record cache support, and customer applications can use it too.
- Caches can be created in 64-bit or recoverable system heap.
- Logical record caches have a fixed number of entries.

Logical record cache definitions

- Logical record cache definitions are a “blueprint” for the cache where relevant attributes can be defined before creating the cache.
- Use cache definitions to set attributes like number of entries, entry size, and expiration time.

If a cache is too small…

- When you create a cache, you need to consider the number of entries that the cache will be able to hold.
- If the cache does not contain enough entries for your workload…
  - Entries may be cast out from the cache before they can be used.
  - Apps may need to fetch data from DASD, or recalculate values, rather than relying on cached data.

Sizing a logical record cache

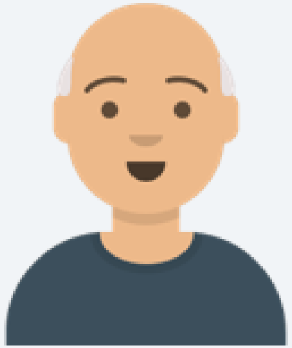
- A properly sized cache should have…
  - …very few (if any) entries not in-use.
  - …very few valid entries cast out.
  - …a hit ratio around 80% or higher.
- Check these stats with **ZCACH DISPLAY COUNTS** and data collection/data reduction.

Problem statement

If a cache is too small, then it isn’t being used to its full potential.

As workloads change, you may need to increase the size of a cache while the z/TPF system is running. As a result, the cached entries are lost.

Users

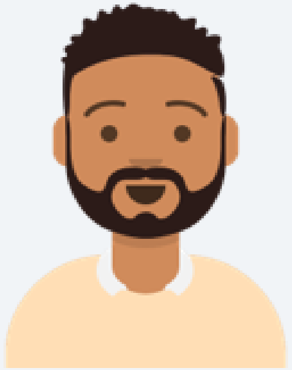


Calvin
Capacity planner

As a capacity planner, Calvin ensures that the z/TPF system’s resources are allocated appropriately for the system’s workload.

Among other types of resources, Calvin is responsible for sizing caches so that applications can use cached data consistently.

Users



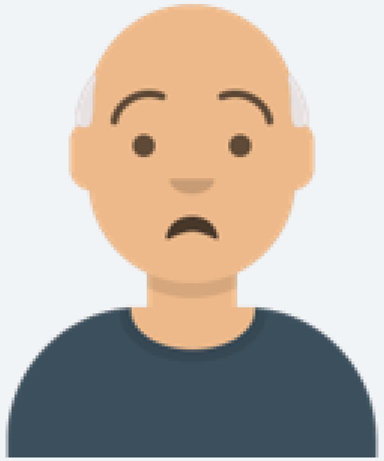
Zach
Application developer

Zach, the application developer, cares about the efficiency of his applications.

Zach wants to be sure that his applications are using cache support efficiently, and that TPF's cache support is performant enough to handle any workload.

Pain points: Capacity planner

- As workloads change, the appropriate size for a cache might change.
- Deleting and re-creating a cache has an impact on applications; will need to be scheduled and approved.
- Online displays of cache entry counts don't account for expired entries; it's difficult to tell if the cache is sized appropriately.



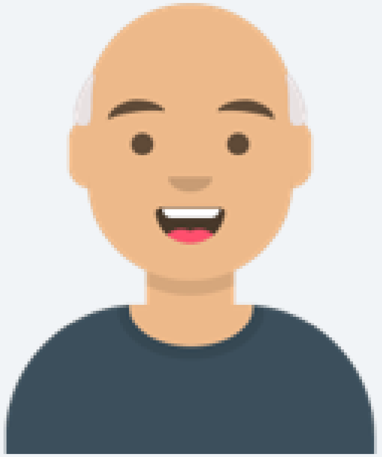
Value statement: Capacity planner

A capacity planner can increase the size of a cache while applications are using it, without losing any of the existing cached data or spending resources repopulating the cache from scratch.

A capacity planner can rely on displays of cache entry counts to accurately represent current cache usage statistics, in order to make informed decisions on cache sizing.

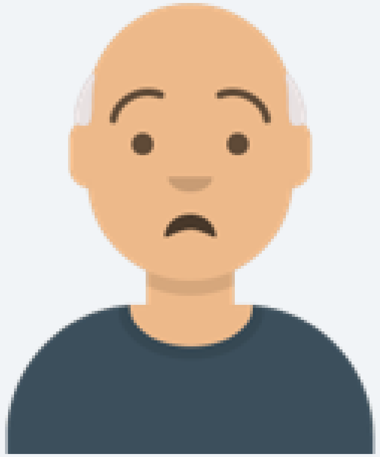
As-is: Capacity planner

- First, Calvin specifies that a certain cache should contain 50 million entries.
- Calvin creates the cache and applications begin to populate it.
- The cache is appropriately full – as entries expire, applications populate the cache with new entries.



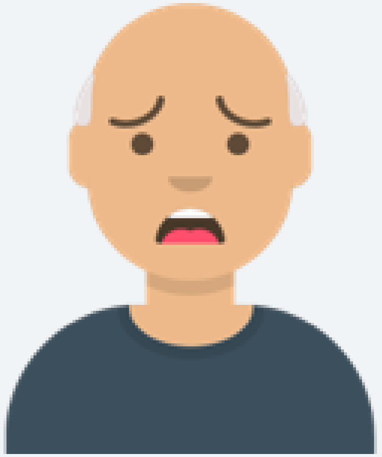
As-is: Capacity planner (continued)

- Over the course of 6 months, traffic to this z/TPF system increases in volume.
- Entries are being cast out more quickly than they can be used.
- Calvin changes the `DATACACHE` cache definition so that it can hold 150 million entries.



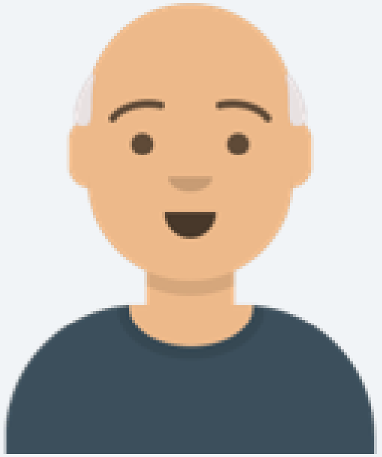
As-is: Capacity planner (continued)

- In order to resize the cache, Calvin has to delete and re-create the cache.
- Calvin goes through a lengthy approval process before he can delete the cache.
- When the cache is deleted, all of the data in the cache is lost.
- While the cache is being repopulated, resource utilization goes through the roof.



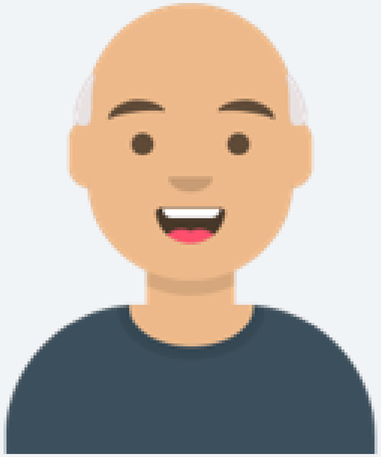
To-be: Capacity planner

- First, Calvin specifies that a certain cache should contain 50 million entries.
- Calvin also specifies that this cache could contain up to 250 million entries in the future.
- Apps run smoothly with the cache at 50 million entries for some time.



To-be: Capacity planner (continued)

- Over the course of 6 months, traffic to the z/TPF system increases in volume.
- With the increase in traffic volume, Calvin resizes the cache so that it can hold 150 million entries.
- The cache can store more data, has room to grow in the future, and doesn’t need to be populated from scratch.



Technical details: Resizing caches

- Existing cache definitions have a `numEntries` attribute.
  - Controls the number of entries in the cache
  - To change the value in the active cache today:
    - Change `numEntries` in the cache definition
    - Delete the cache
    - Re-create the cache

Technical details: Resizing caches (continued)

- New `maxNumEntries` attribute for cache definitions
  - Controls the maximum number of entries that the cache will ever be able to hold
  - To resize a cache without losing data:
    - Increase the value of `numEntries`
    - Enter `ZCACH REFRESH` *`cacheName`*
  - Must specify `maxNumEntries` to resize a cache

Example: Resizing caches

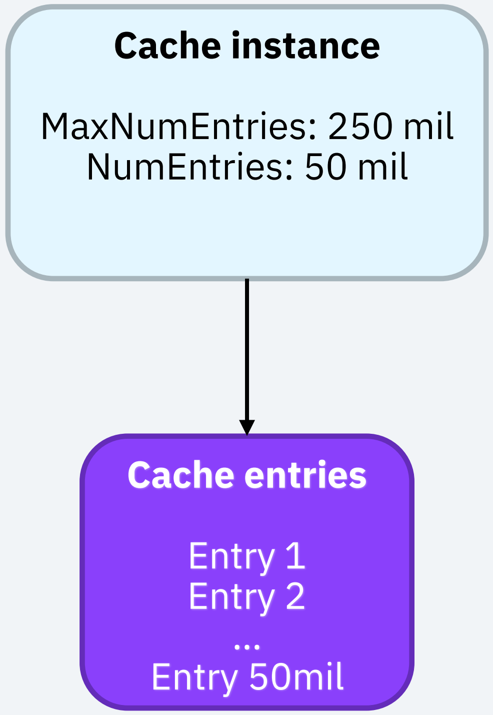
Number of entries in cache defn	Max number of entries	Actual number of entries	Hash table size	Total system heap used
50 million	250 million	50 million	~5.5 GB	~25GB
150 million	250 million	50 million	~5.5 GB	~25GB
150 million	250 million	150 million	~5.5 GB	~63GB

Operator enters
`ZCACH DEFINITION ALTER`

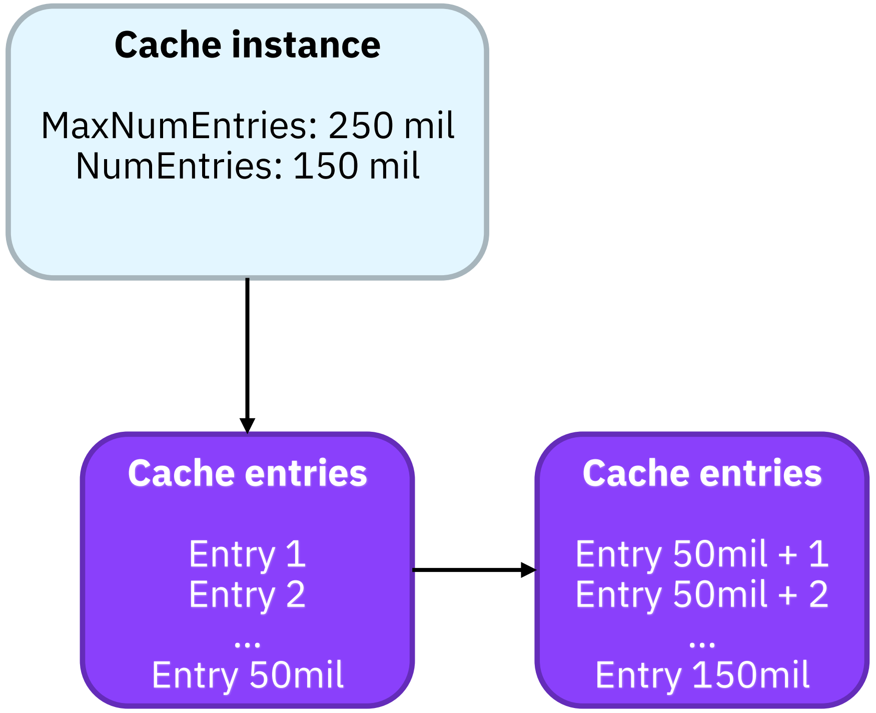
Operator enters
`ZCACH REFRESH`

Number of hash table entries ≈ (maxNumEntries / 2)
Key length = 16
Data length = 256

Before refresh

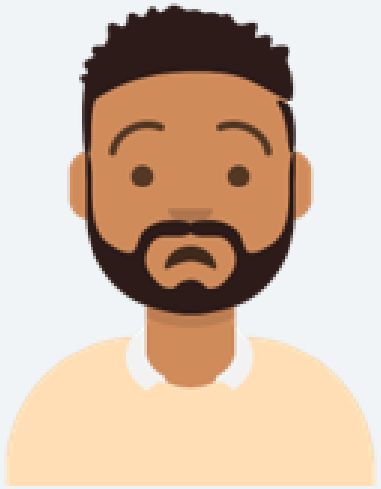


After refresh



Pain points: Application developer

- Adding an entry to a very large, full cache can take a lot of time.
- Cache support uses application ECB time to search for expired entries when the cache is full.

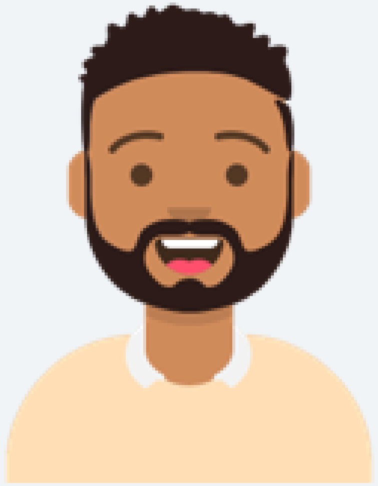


Value statement: Application developer

An application developer can expect that application performance will not be impacted by adding entries to a logical record cache.

To-be: Application developer

- A system-controlled background process prunes expired entries from all caches on the system.
- When Zach’s app adds an entry to a cache, there is room for the new entry.
- Zach doesn’t need to worry about the performance impact of searching a full, large cache.



Technical details: Cache sweeper

- New cache sweeper process
  - Entirely IBM-controlled
  - Removes expired entries from caches without using application ECB time
  - Requires castout time (`CTIME`) attribute to be set

Technical details: Internal entry management

- Today:
  - Cache support may search through all cache entries looking for an expired entry to cast out.

- With this enhancement:
  - Cache support will search through a **limited** number of entries looking for an expired entry to cast out.
- This enhancement applies to **all** caches, resizable or not.

Requirements

- Must use cache definitions to resize a cache
- Number of entries can’t be dynamically decreased
- Cache must be processor unique with an entry size less than 4096 bytes to be resized
- To increase the maximum number of entries, delete and re-create the cache

Conclusion

- Use cache definitions to increase the number of entries in a cache without losing the data in the cache.
- Updating entries in all caches (including system-managed caches) will be more efficient.
- Console displays of cache utilization will more accurately represent the number of usable entries in the cache.

Thank you

Appendix

Converting caches to use cache definitions

- There are two ways to create and manage caches.
  - `newCache()`, `tpf_newCache_ext()`, and `deleteCache()` APIs (legacy support)
  - Cache definitions (introduced in 2019)
- Caches can only be resized if they're managed by a cache definition.
- To convert API-managed caches to cache definitions, refer to the "Manage logical record caches and logical record cache definitions" topics in IBM Documentation.