Adabas Caching Facility

Adabas Caching Configuration and Tuning

Version 8.2.2

May 2011

Adabas Caching Facility

Table of Contents

1 Adabas Caching Configuration and Tuning

This document provides information related to Adabas Caching configuration and performance tuning.

Adabas Caching Runtime (ADARUN) Configuration Parameters

Adabas Caching configuration and execution is controlled by the settings defined in the Adabas Caching ADARUN parameters.

For a complete description of these parameters, see the section Adabas Caching Parameters.

Adabas Caching Configuration

Caching Level

Adabas Caching provides for caching RABNs globally by RABN range, or at the file level by caching all RABNs associated with a particular file or range of files.

File-level caching and global caching for Associator and Data Storage are mutually exclusive in that only one or the other mechanism may be used.

If the ADARUN parameter `CACHE=YES` is specified and neither RABN ranges nor files or file ranges are specified in the ADARUN parameters (`CFILE` or at least one of the parameters `CSTORAGE`, `CASSOxxx`, or `CDATAxxx`), no caching occurs. Instead, ADACSH defaults to a global caching mode that allows RABN ranges to be added dynamically using operator or Online Services commands. In this situation, any attempt to start caching on a file level is rejected.

The RABNs on the Work datasets are not associated with any file; thus, Work caching is always a global process. Only the Work caching specified using ADARUN parameters (`CWORKSTORAGE`, `CWORK2FAC`, `CWORK3FAC`) is possible during an Adabas session. There is currently no way to dynamically define a Work part 2 and/or 3 RABN range.

Global Caching

Global caching refers to the process of caching a RABN as part of a range of RABNs. A number of different RABN ranges can be specified. They are then cached at the appropriate time when encountered during runtime.

Global caching can be used to target heavily-used RABNs explicitly. However, the RABN ranges specified must be monitored regularly as this mechanism can be disrupted by database reorganization and/or changes in the profile of the work load.

File-Level Caching

File-level caching refers to the process of caching RABNs based on the file with which they are associated.

A file is a candidate for caching when it impacts the performance of the system, but its working set cannot be maintained in the Adabas buffer pool for some reason. The RABNs belonging to such a file are read continually from disk, indicating that they are in high demand but not so high that they remain in the Adabas buffer pool.

When a file is cached in the ADACSH cache area, its RABNs are brought back into the buffer pool without any I/O, enabling transactions that use the cached file to complete quickly.

Caching a RABN in a file minimizes maintenance if the database or file is reorganized. Only file usage needs to be monitored, since ADACSH adjusts automatically its RABN ranges to any changes in the structure or location of the file's RABNs. The RABN is still associated with the file number and therefore, is still cached after the reorganization.

The `CFILE` parameter identifies one or more files for which file-level caching is to be activated.

Before the Adabas nucleus purges a RABN from its buffer pool, ADACSH is called to determine whether the RABN is to be cached. The file number with which the RABN is associated is located in its buffer pool header. ADACSH checks the list of files for which caching is active and, if RABNs are being cached for the file, copies the RABN to cache storage for retrieval by the Adabas nucleus when required. Retrieval depends on the appropriate part of the file being cached.

Caching Scope

Caching for Associator RABNs, Data Storage RABNs, and Work RABNs may be specified:

- for global caching, different ADARUN parameters (and operator commands) are available to specify caching for Associator RABNs (`CASSOxxx`) and Data Storage RABNs (`CDATAxxx`).
- for file-level caching, the `CFILE` parameter is used to select the RABN types to be cached.
- RABNs for Work parts 2 and 3 are specified for caching using the `CWORKSTORAGE`, `CWORK2FAC`, and `CWORK3FAC` parameters for both global and file-level caching.

Example:

You might choose to cache Associator RABNs but not Data Storage RABNs if the activity against Associator RABNs in a file is very high due to constant searching:

- if the size of the file is such that these Associator RABNs consistently remain in the Adabas buffer pool, you may want to make only the Data Storage elements available in the ADACSH cache if they do not exist in the Adabas buffer pool.
- if the file is quite large and the Associator RABNs are read continually into the buffer pool for such searches, you could considerably lessen the impact of these searches by caching the Associator RABNs so that at least the Associator RABN read would not result in an I/O. With such a large database, different data elements would probably be referenced each time. If so, it would not make sense to waste cache space caching Data Storage RABNs that would only be referenced once.

Caching Space Types

A cache space, as used by ADACSH, is a *logical* amount of memory available for caching a specific type of Adabas RABN in a specific type of memory or cache.

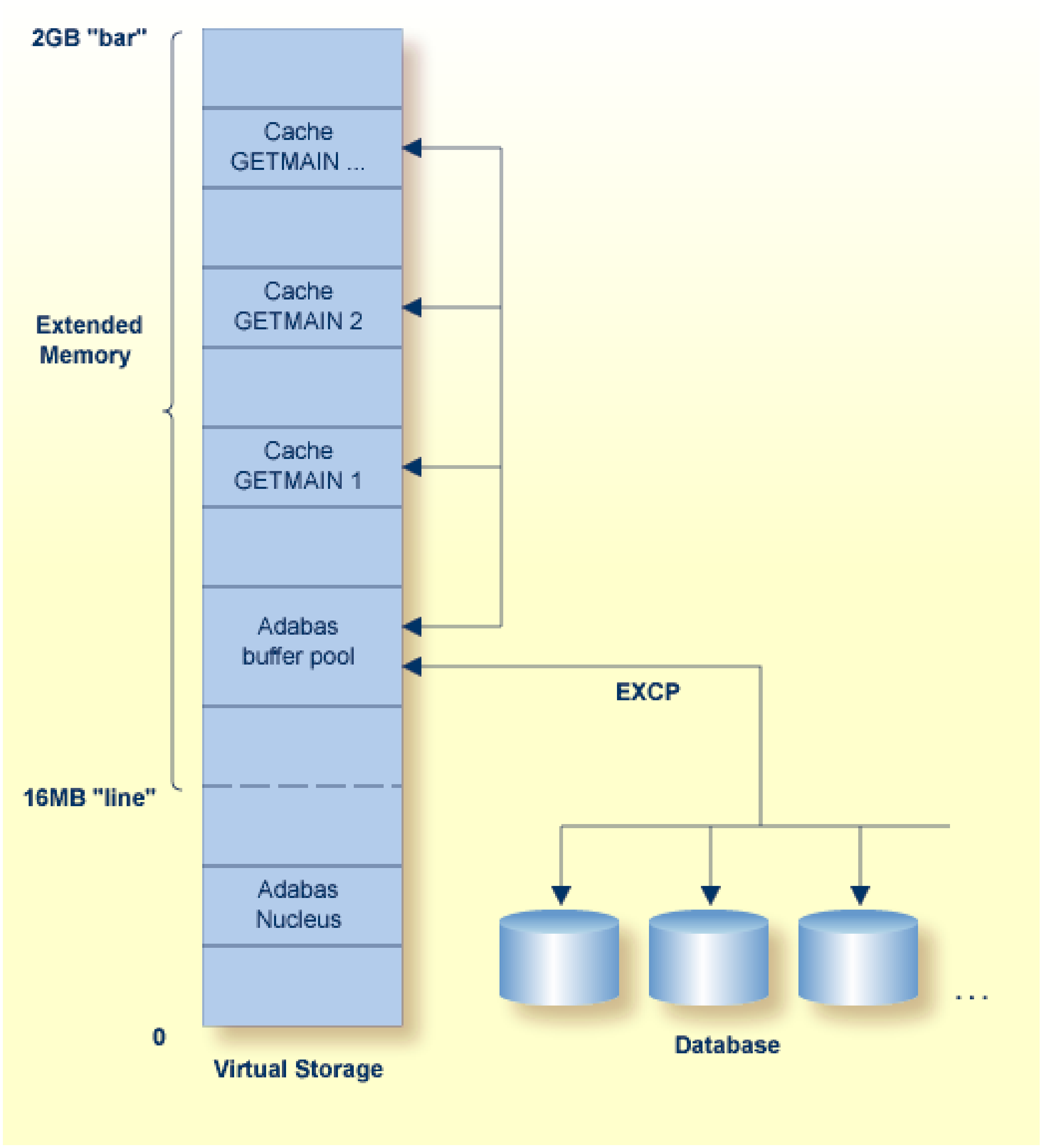
Cache spaces can be defined in extended memory, data spaces, hiperspace, virtual 64, and virtual 64 backed by large pages storage. Once defined, cache areas are allocated dynamically as RABNs move to and from the cache areas and to and from the Adabas buffer pool.

The type of cache to use depends on the environment in which caching is to be used.

- **Extended Memory**
- **Data Spaces**
- **Hiperspaces**
- **64-Bit Storage backed by large pages**
- **64-Bit Storage**

Extended Memory

If a relatively small amount of cache space is required, it may make sense to allocate this space within the extended memory area available to Adabas. This is the fastest option for caching. It is currently the only option available for BS2000 RISC machines.

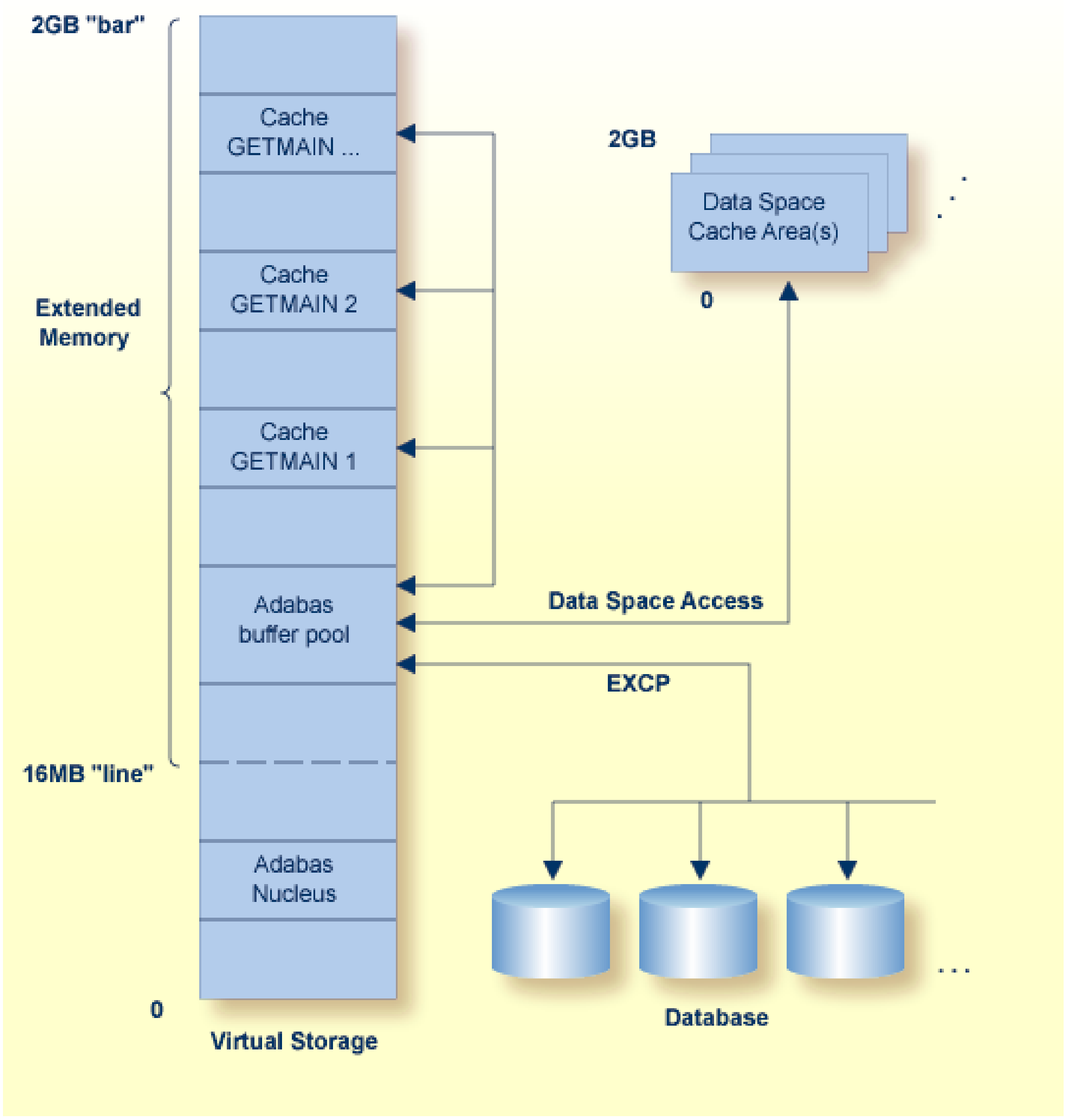


Extended Memory Caching

Data Spaces

Where a large amount of cache space is required, it may be necessary to allocate the cache space in data spaces. This option is slightly slower than using extended memory and is available in BS2000 (except RISC machines) and ESA operating environments (VM/ESA, VSE/ESA, MVS/ESA, and z/OS platforms).

The following graphic shows cache space allocated in both extended memory and data spaces:

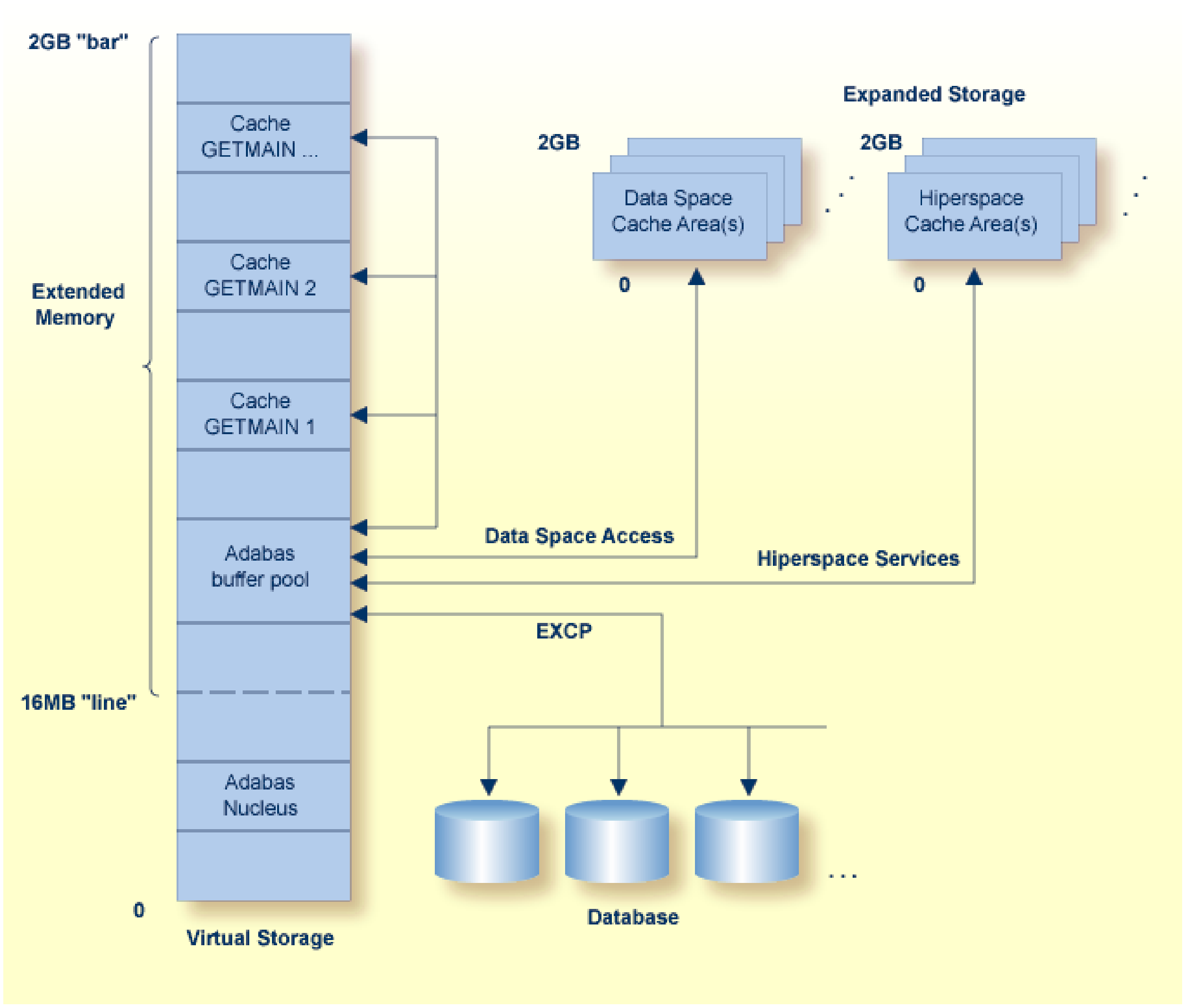


Caching in Extended Memory and Data Spaces

Hiperspaces

It is possible to allocate cache spaces in hiperspace. This option is relatively slow compared to data space or extended memory access but is still significantly faster than disk access. This option may be useful when the use of data spaces or extended memory would cause too much paging on the system.

The following graphic shows cache space allocated in extended memory, data spaces, and hiperspace:

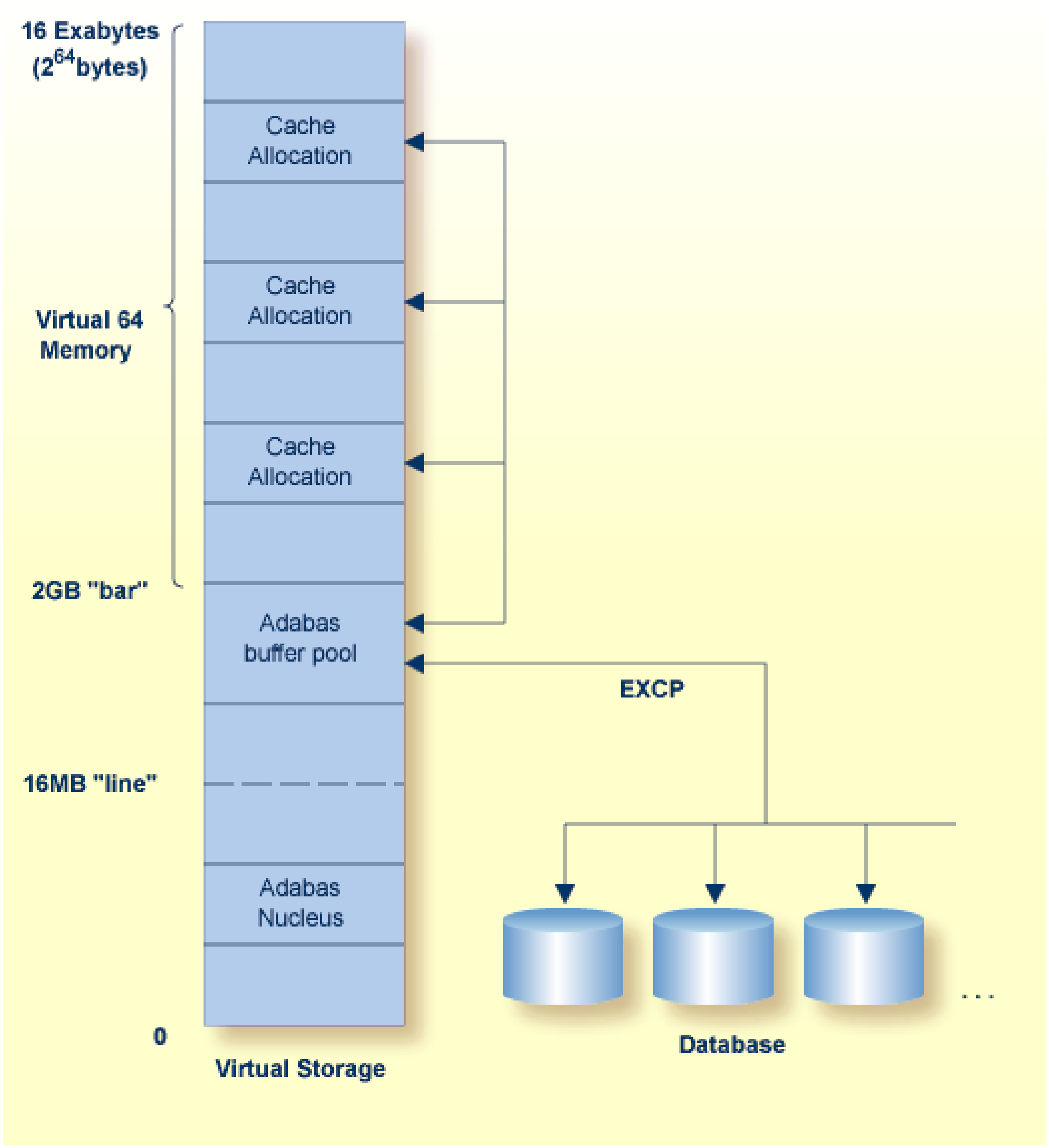


Caching in Extended Memory, Data Spaces, and Hiperspace

64-Bit Storage and 64-Bit Storage backed by large pages

In z/OS environments when a large amount of cache space is required, it is possible to allocate cache spaces in 64-bit virtual storage and 64-bit virtual storage backed by large pages.

The following graphic shows cache space allocated in virtual 64 memory:



Caching in Virtual 64 Memory

Above the 2GB bar in 64-bit virtual address space, cache spaces are allocated in chunks of virtual storage called memory objects, each of which represents a number of virtual segments, each of those a megabyte in size and beginning on a megabyte boundary. The maximum size of memory objects is controlled by your installation: a memory object can be as large as the memory limits set by your installation and as small as one megabyte.

For information about 64-bit address space, refer to the IBM documentation *SA22-7614-00, MVS Programming: Extended Addressability Guide, second edition October 2001*, which applies to z/OS 1.2 and above. For information about how your installation has implemented its use of 64-bit address space, contact your system programmer.

Number and Size of Cache Space Areas

The maximum size of the storage that may be allocated for a given cache space is determined by the `CMAXCSPS` parameter and the `CASSOMAXS` and `CDATAMAXS` parameters:

```
CMAXCSPS · CDATAMAXS
or
CMAXCSPS · CASSOMAXS
```

When space is required for a RABN range or file, ADACSH allocates an area of storage in the type of cache requested (extended memory, data space, hiperspace, virtual 64 storage or virtual 64 storage backed by large pages) with a size of `CASSOMAXS` or `CDATAMAXS`, depending on the type of RABN being cached. When this space is exhausted, another area of storage is allocated until the maximum number of areas has been allocated as specified by `CMAXCSPS`.

Note: Cache spaces for Work RABNs are allocated in one block according to the Work caching parameters (`CWORKSTORAGE`, `CWORK2FAC`, `CWORK3FAC`).

How Cache Space is Used (Class of Service)

When global caching is active, all of the cache space associated with a given RABN range is available for caching RABNs within that range.

When file-level caching is active, the amount of space that can be used for caching RABNs associated with a file is determined by the class of service for the file. The class determines the maximum amount of a cache space that the file can use and when the space will be reused.

Class of service effectively prioritizes a file within the caching subsystem. The higher the priority of a file, the more space it has available when it is in use. When a higher-priority file is not in use, the space is available to lower-priority files until such time as a higher-priority file is again in use.

ADACSH has implemented 5 classes of service numbered 1 to 5 where 1 is the highest priority and 5 is the lowest priority:

Class	Percentage
1	up to 100% (highest priority)
2	up to 75%
3	up to 50% (the default)
4	up to 25%
5	up to 10% (lowest priority)

The percentage relates to a given file with a given class of service and *not* all files with that class of service. For example, if 6 files with class 5 (up to 10% of available cache space) are in use in the system, as much as 60% (6 · 10%) of a logical cache space could be used.

If both Associator and Data Storage RABNs are being cached for a file, the file will have the defined percentage available in each of the relevant cache spaces: ASSO and DATA. For example, if a file with class 3 is caching both Associator and Data Storage RABNs in data space, as much as 50% of the available Associator data space cache and 50% of the Data Storage data space cache could be used by the file.

If ASSO/DATA cache spaces are allocated in extended memory and in data space, the percentage only applies to the type of cache space where the RABNs are being cached.

Releasing and Reusing Cache Space

Cache space can be released for reuse in two ways:

- **Inactivity**
  Using the `CCTIMEOUT` parameter, cache space areas can be released or RABN ranges disabled based on a specified period of inactivity. If the `CDEMAND` parameter is also used, disabled RABN ranges are automatically enabled when the demands on the Adabas buffer pool increase beyond a specified level.
- **Age**
  When a RABN is to be moved from the Adabas buffer pool to the ADACSH cache space and no cache space is available and the maximum amount of space has already been allocated, ADACSH makes space by purging the RABN that has been in the cache space for the longest time and reusing that space for the new RABN to be written to the cache. The purged RABN must then be reread from disk when needed by the nucleus. This process essentially reflects a first-in-first-out (FIFO) mode of operation.

Note: If the Work datasets are cached 100%, no aging is performed as all Work blocks can fit in the available space.

Inactivity Timeout

For each individual area of storage allocated for a logical cache space, a record is kept of the last time this area of storage was referenced. When the difference between the current time and the last referenced time is greater than the specified `CCTIMEOUT` value, the cache space is released. All RABNs cached in the released space must be read again from disk if required.

When demand-level caching (`CDEMAND` parameter) is active, a record is kept of the last time a given RABN range was referenced. When the difference between the current time and the last referenced time exceeds the `CCTIMEOUT` specification, the RABN range is disabled. This applies for global RABN ranges and RABN ranges associated with a file. All ranges are enabled again automatically when the Adabas buffer efficiency falls below the level specified using the `CDEMAND` parameter.

The Aging Process

If a RABN is being moved from the Adabas buffer pool to the ADACSH cache space and no space is available, and the maximum amount of space has been allocated, ADACSH must make space.

This is done by remembering the oldest RABN in the cache space. The oldest RABN is the RABN that has been in the cache space for the longest time. This RABN is purged and the space it occupied is reused for the new RABN to be written to the cache. The purged RABN must then be reread from disk if required at a later time. This essentially reflects a first-in-first-out (FIFO) mode of operation.

File-level caching uses the same theory as global caching in that it attempts to reuse the oldest blocks in the cache space when the cache space is full. This is refined slightly by the file's class of service assignment.

When file-level caching is active, five aging chains are maintained, each of which contains the list of RABNs in the cache associated with a given class of service. Within the aging chain for each class of service, ADACSH also maintains a chain of the oldest elements for each file cached with that service class.

Suppose a logical cache space is full (that is, all cache space extents have been allocated and there is no free space in any extent) and a new RABN arrives to be cached.

- If the new RABN is to be cached for a file that is already using the maximum amount of space allowed to it based on its class of service, the oldest RABN in the cache space for that file is purged and the space released is used by the new RABN.
- If the new RABN is to be cached for a file that can still use more space based on its class of service, the oldest RABN in the lowest priority service class is purged from the cache space to accommodate the new RABN. In other words, the RABNs on the class 5 aging chain are reused first; class 4 second; and so on. This insures that the highest priority class 1 files remain in the cache the longest.

For this aging mechanism to be effective, there must be a mix of files with a good mix of service classes. If all files have the same or high priority service classes, the effects of this mechanism will not be as pronounced. If all files are class 1, the oldest RABN in the cache will simply be purged each time. There may be some environments where this is a desirable situation.

Performance and Tuning

This section provides information related to Adabas Caching performance and tuning. The user's environment and job mix influence how the Adabas buffer pool is used which, in turn, determines the effectiveness of the dynamic cache.

Sizing the Buffer Pool and Cache Space Areas

The key factor in achieving maximum efficiency is the size of the Adabas buffer pool (`LBP`) along with the sizes allocated for the cache space areas. Performance gains may be realized by providing a sufficient number of cache space areas (`CMAXCSPS`) to accommodate the database's working set; that is, the set of Associator and Data Storage blocks accessed during the Adabas session. This effectively reduces the physical read I/O to one EXCP per block used.

A usable cache space area size (`CASSOMAXS` and `CDATAMAXS` parameters) in conjunction with the Adabas buffer pool size (`LBP`) may not be so easily determined. Each installation should experiment to find the best parameter settings, using as a guide the following operator commands to display information about the caching performance:

- `CSTAT` (display cache statistics for RABN ranges or Work parts 2 and 3);
- `CFSTAT` (display cache statistics for files); and
- `CSUM` (display cache summary).

Tuning Parameters for Global Mode Caching

If you used a version of Adabas Dynamic Caching that cached RABNs and RABN ranges only and you continue to cache RABNs and RABN ranges (global mode caching), certain changes in the product require that you tune your parameters when you move to version 6.2 or above.

Prior to version 6.2, a physical cache space area was used for a single RABN range only. Version 6.2 and above caches multiple RABN ranges in a single physical cache space area. For best performance, it is therefore advisable to use fewer, larger cache space areas than were used in previous versions. For example, prior to version 6.2, the default number of cache space areas (`CMAXCSPS` parameter) was 255; for version 6.2 and above, the default number for each RABN type (ASSO, DATA) in each memory-type area (extended, data space, hiperspace, virtual 64) is 8 and the maximum is 16.

Prior to version 6.2, the physical cache space assigned to a RABN range had the size of the RABN range or `C`*`xxxx`*`MAXS`, whichever was smaller. Beyond the maximum number of physical cache spaces (`CMAXCSPS`) for either Associator and Data Storage, definitions were ignored. All physical space needed for caching was allocated dynamically as the need arose.

For version 6.2 and above, the total amount of memory (cache space) available for caching a specific type of RABN (ASSO, DATA) in each type of memory (extended memory, data spaces, hiperspace, virtual 64 and virtual 64 backed by large pages) is logically defined as the product of the size and number of physical storage units that can be allocated for the RABN type (`C`*`xxxx`*`MAXS` times `CMAXCSPS`). Once logically defined, the cache space is dynamically allocated one physical unit (area) at a time as needed.

The `CASSO`*`xxx`* and `CDATA`*`xxx`* parameters indicate the RABN type (ASSO, DATA) and the memory type (EXT, DSP, HSP, L64, V64) and therefore the logical cache space where the RABNs will be stored; for example, all RABNs and RABN ranges specified using the CDATAxxx parameter are cached together in the physical storage areas for Data Storage of the cache space indicated by the parameter suffix.

Note: The `CSTORAGE` parameter can specify a single memory type for all caching and overrides the `CASSO`*`xxx`* and `CDATA`*`xxx`* parameters.

All of the cache space associated with a given RABN range (RABN type and memory type) is available for caching RABNs in that range. When a RABN needs to be cached and no cache space is available and the maximum amount of space has already been allocated, the RABN that has been in the cache space the longest is purged to make room for the new RABN to be written to the cache. The purged RABN must then be reread from disk when needed by the nucleus.

Estimating Read Time Saved by ADACSH

An estimate of the read time saved by ADACSH can be computed by evaluating the statistics for each cache space as follows:

```
read time saved = (AVE EXCPT - AVE NIOT) · CACHE READS
```

The CSTAT and CFSTAT operator commands and/or the nucleus shutdown statistics supply the values for AVE EXCPT, AVE NIOT, and CACHE READS. For further information.

The accumulated read time saved for all cache spaces represents the reduction in I/O processing time afforded by ADACSH. The actual time saved for Adabas is a percentage of the read time saved based on the amount of CPU and I/O overlap that occurs in the environment.

Device Types for ASSO or DATA Storage

The cache space allocated for Associator or Data Storage is based on the block size of that storage. If Associator or Data Storage areas are allocated on differing device types, the block size for the RABN will be different. To avoid complex processing and thus additional overhead, ADACSH simply uses the highest block size in use for the storage in question. If device types with different block sizes are used, valuable cache space is wasted. Therefore, Software AG recommends that you use only devices with the same block size when operating with ADACSH.

Running Adabas Utilities

When an Adabas utility such as ADADBS, ADAINV, ADALOD, ADAORD, or ADARES runs as a UTI user while the nucleus is active, the active Associator and Data Storage caching areas are released to preserve database integrity because the utilities can modify the DASD of the database directly. Similarly, the nucleus releases the necessary blocks from the Adabas buffer pool.

Once the utility function is complete, the caching areas are reactivated as necessary; however, because of the physical I/O required to reread blocks into the buffer pool, performance may be degraded if a large number of parallel utility operations is constantly run against heavily accessed or updated files.

Enhancing Hiperspace Operation with Move Page

The move page (MVPG) operation under z/OS reduces the linkage and checking incurred by the operating system when moving pages to and from expanded storage (hiperspaces). Because the benefits are greater when a program moves relatively small amounts of data frequently, the MVPG should enhance the ADACSH hiperspace operations.

ADACSH is designed to automatically detect whether the MVPG facility is installed. No parameters are required to enable or disable this feature.

Extended Memory Considerations

If you have a limited amount of extended memory (above the 16MB line) for use by ADACSH, *do not* allow ADACSH to use the GCB RABN ranges as the default (that is, by specifying `CACHE=YES`, `CSTORAGE=storage`). This is especially important when there are large numbers of unreserved RABNs in the Adabas free space table. The reason is that, for each RABN available on the database, 4 bytes is required for its RABN table entry.

Consider caching only the appropriate file to limit the resources allocated to just those required for the number of RABNs actually used in a file's Data Storage or Associator storage plus 10% for expansion. The RABN tables are expanded automatically if more RABNs are used for the file. In this way, extended memory is more efficiently used.

If global caching is used, that is the RABNs to be cached are specified directly, consider the number of RABNs cached relative to the type of cache space. For example, it might be more efficient to cache a smaller number of RABNs in extended memory rather than in a data space or hiperspace because extended memory access incurs less overhead than access to a data space or hiperspace.

Estimating Memory Space Requirements

Adabas Caching Facility allocates memory in chunks based on the `CASSOMAXS` and `CDATAMAXS` definitions. The number of chunks to be allocated is determined by the CMAXCSPS definition.

The memory space requirements for ADACSH may be estimated by using the formula described in the following sections:

- Space Requirements for both Global and File-Level Caching
- Space Requirements for Global Caching
- Space Requirements for File-Level Caching

Space Requirements for both Global and File-Level Caching

The following are the space requirements for both global and file-level caching.

Cache Main Control Block

```
CMCBL + 16,384
```

where:

CMCBL	is the length of the cache main control block (4096)
16,384	is used for working storage

Cache Storage Control Blocks

```
( CSCBL + ( CSCEL · mcs ) ) · ncb
```

where:

CSCBL	is the length of the cache storage control block (208)
CSCEL	is the length of the cache storage control extent block (160)
mcs	is the maximum cache spaces
ncb	is the number of cache areas allocated

RABNINDX Control Block

```
RABNINDL · ( tes ÷ bs )
```

where:

RABNINDL	is the length of the RABNINDX control block (64)
tes	is the total amount of cache storage allocated
bs	is the block size for the RABNs being cached

IONRLST Control Block

```
( ( maxthreads + 2 ) · IONRLSTL ) + 32
```

where:

IONRLSTL	is the length of the IONRLST control block (40)
maxthreads	is the number of Adabas threads defined
32	is the number of bytes used to ID the storage area

TRKBUFCB Control Block

```
( ( maxthreads + 2 ) · TRKBUFLL ) + 32
```

where:

TRKBUFLL	is the length of the TRKBUFCB control block (40)
maxthreads	number of Adabas threads defined
32	is the number of bytes used to ID the storage area

Operator Communication Area

```
256 + 32
```

where:

256	is for the operator communication area
32	is the number of bytes used to ID the storage area

Number of Track Input/Output Buffers

```
bufno · ( MAXTRKSZ + 256 + 32 )
```

where:

bufno	is the number of track I/O buffers specified
MAXTRKSZ	is the (DATA/ASSO blocksize + gapsize) blocks track + 256
32	is the number of bytes used to ID the storage area

Space Requirements for Global Caching

FNRLIST Control Block

The following is allocated at initialization for global caching:

```
FNRLISTL · mfn + 32
```

where:

FNRLISTL	is the length of the FNRLIST control block (16)
mfn	is the maximum number of files on the database
32	is the number of bytes used to ID the storage area

For Each Global Extent Cached

The following are the space requirements for *each* global extent being cached.

```
CREBL
```

where:

CREBL	is the length of the cache RABN extent block (160)

```
CRTPL + ( 8 * nnr )
```

where:

CRTPL	is the length of the cache RABN table prefix (16)
nnr	is the number of RABNs in the RABN range

Space Requirements for File-Level Caching

Cache File Table Prefix

The following are the space requirements when file caching is active, regardless of how many files are being cached:

```
CFTPL + ( 4 · mfn )
```

where:

CFTPL	is the length of the cache file table prefix (16)
mfn	is the maximum number of files in the database

For Each File Cached

The following are the space requirements for *each* file being cached.

```
CFCBL
```

where:

CFCBL	is the length of the cache file control block (256)

```
CREBL · ( nae + nde )
```

where:

CREBL	is the length of the cache RABN extent block (160)
nae	is the number of ASSO extents
nde	is the number of DATA extents

```
CRTPL + ( 4 · nrr )
```

where:

CRTPL	is the length of the cache RABN table prefix (16)
nrr	is the number of RABNs used in the file at startup + 10%