Cache information and management

By Thomas SCHLUMBERGER, Technical Support Engineer, 4D France.

Technical Note 25-02

Table of Contents

Abstract

The cache in 4D is a buffer that stores created, modified, or frequently queried data. It operates as a reserved part of the 4D Engine, but to optimize its performance, certain settings and commands are available to interact with the cache. Cache performance is influenced not only by the server configuration but also by the design of the database.

Introduction

After providing a brief history of the evolution of cache settings in 4D and explaining how cache handling is managed by the 4D Engine, optimized regardless of whether its size is fixed or calculated, the Technical Note focuses on possible interactions to improve cache performance. Certain settings change can be applied to adjust the cache's size and flush frequency, and commands are available to interact with its content.

The Technical Note explains what the cache is, how it has evolved, and what tools are available to interact with it.

Requirements

The following versions of 4D/4D Server are required for the Technical Note:

- 4D 20.5 LTS or later version (20 R5, 20 R6, 20 R7, etc….)

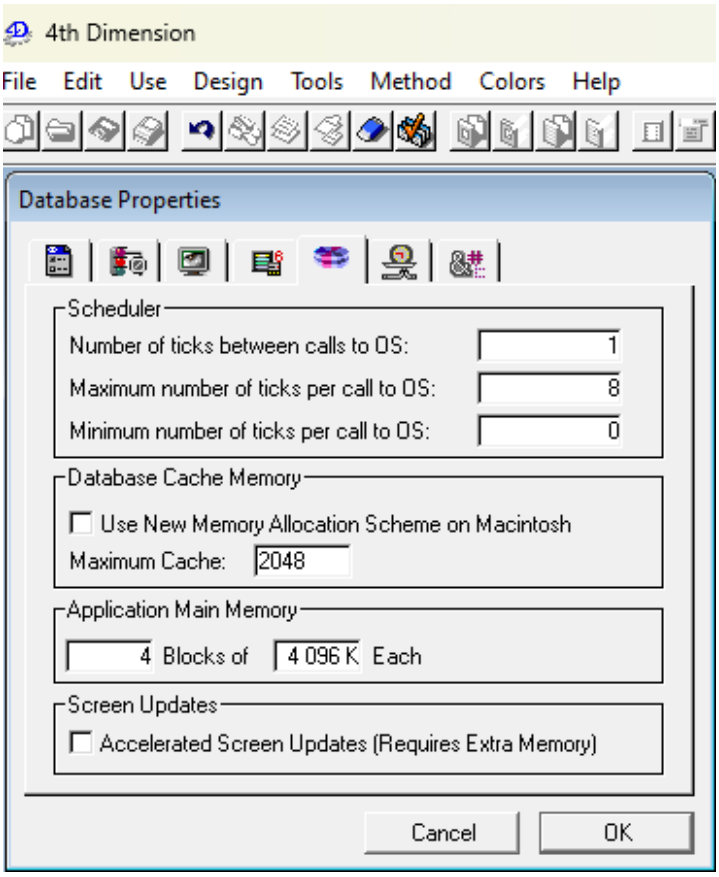
A brief history of the Cache settings in 4D

Cache has always existed in 4D, as it is the optimal intermediate way to access structure and data from the files. Until recently, connected hard disk(s) were slow compared to SSD, and with small capacity. 4D was directly managing memory space for the cache set for the database.

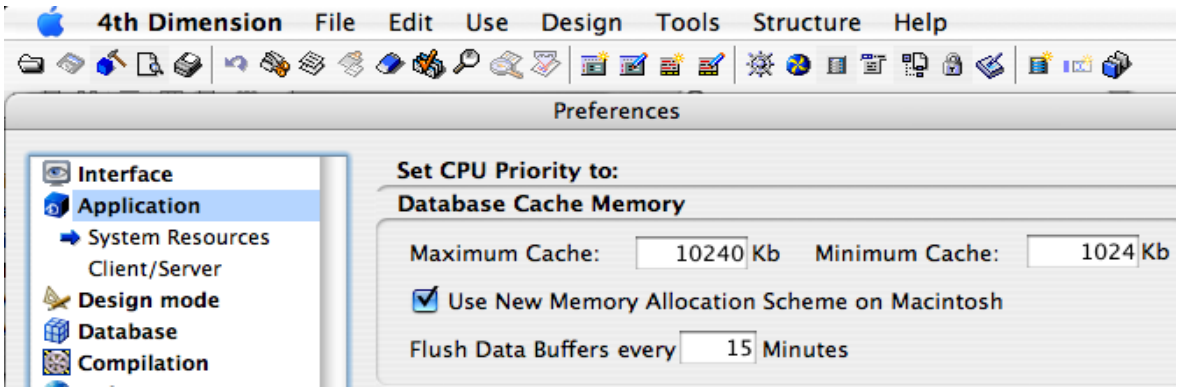
In version 6.0, the Cache settings were very limited, mainly the Maximum Cache:

Cache has always existed in 4D, as it provides the optimal intermediate way to access structure and data from the files. Not too long ago, connected hard disks were much slower compared to SSDs and had smaller capacities. In earlier versions, 4D directly managed the memory space for the cache set for the database.

In version 6.0, the cache settings were very limited, mainly to the Maximum Cache size:

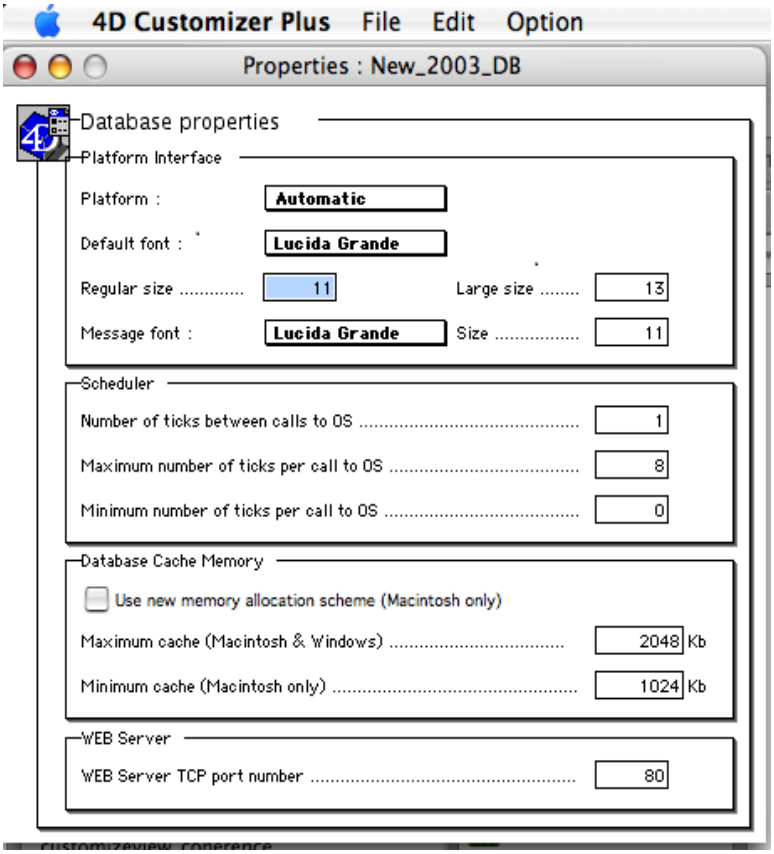


In version 2003, the Flush Data frequency setting was introduced as new feature:

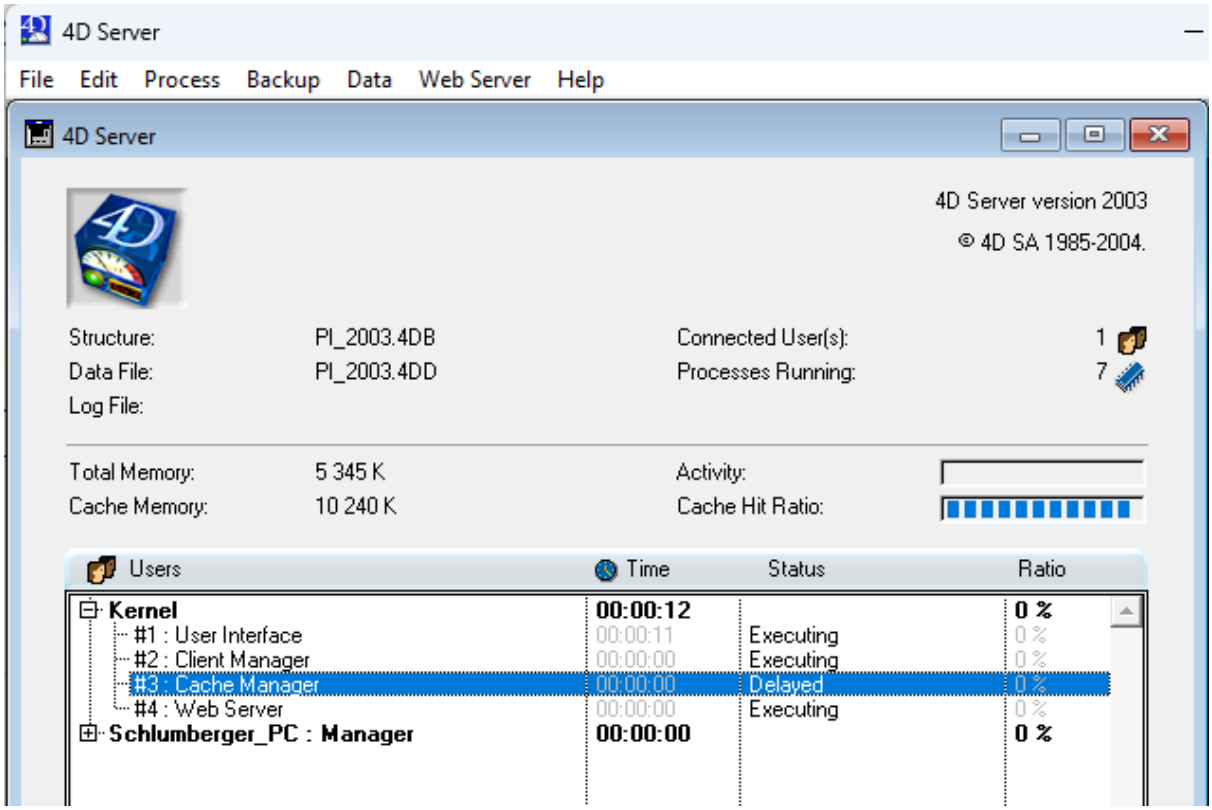


In versions 5.x, 6.0, 6.x, and up to 2004, 4D Customizer could be used to set basic settings for the Cache (Database Properties) of a structure file. In the following example, using the 4D 2003 version on Mac OS X:

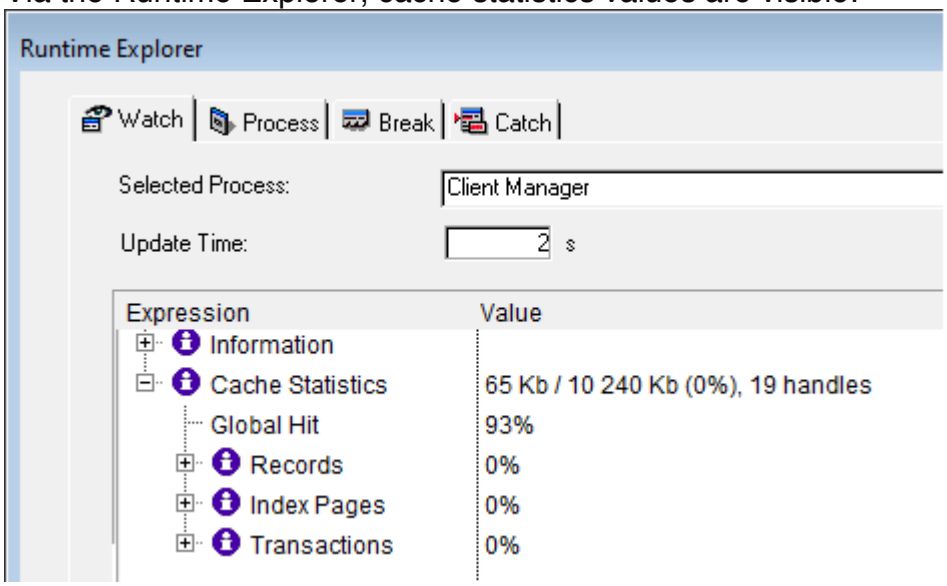
The list of 4D Customizer properties:



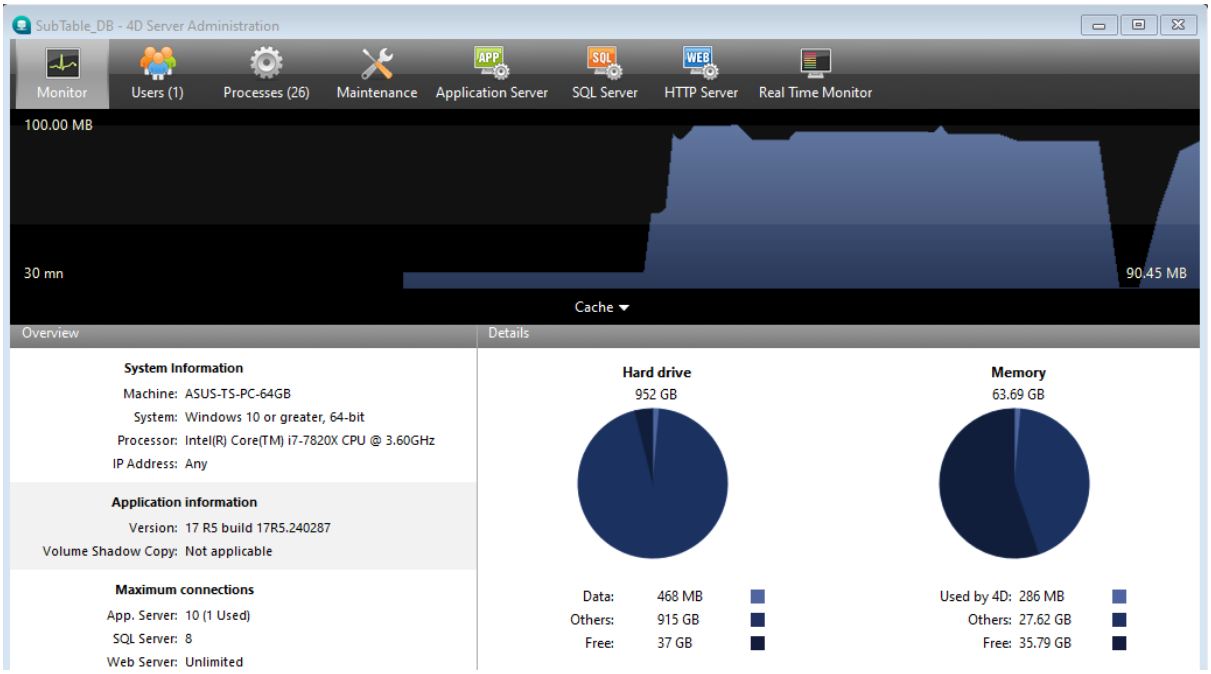
For the monitoring of the Cache of 4D Server, the display of information was limited. For example, with 4D 2003, no historical graphics were available:

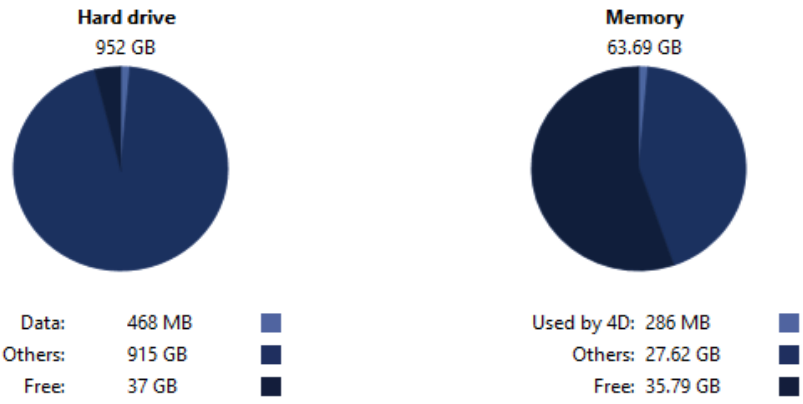


Via the Runtime Explorer, cache statistics values are visible:



Starting with **version 11**, the new Administration window is significantly enhanced, with, for example, a graph on top showing Cache usage (The current Cache size is 100.00 MB):



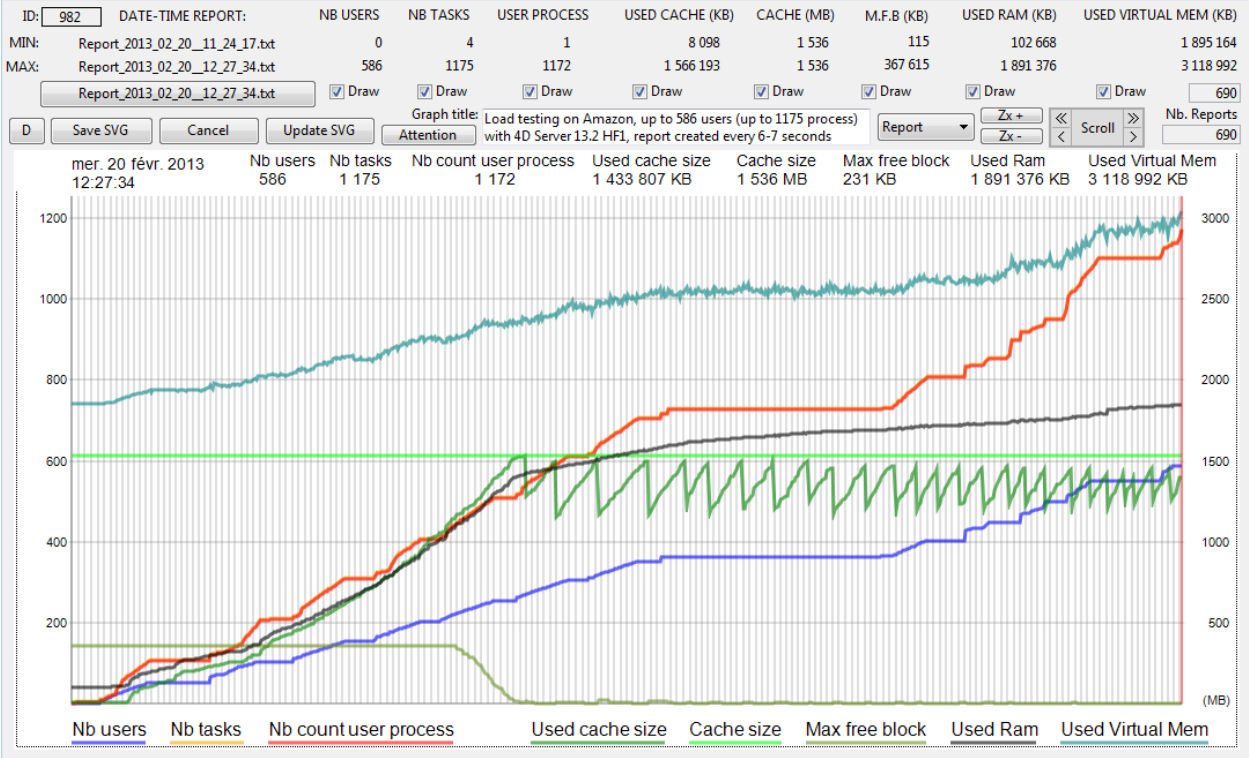


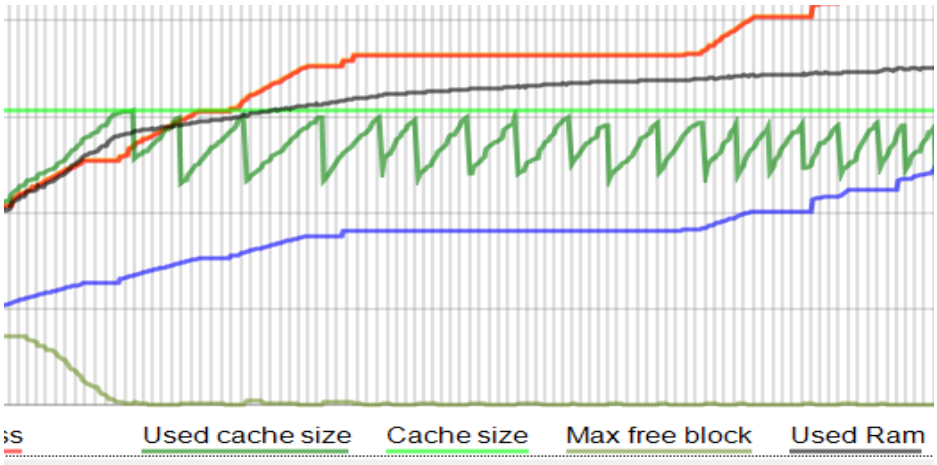
Note: The interval in the graph grows from 15 minutes to 4 hours, depending on how long the database has been running.

Before 4D/4D Server v16 (64-bit)

The Cache management was using blocks of free memory space.

When a room was required, a quarter of the Cache was flushed, as shown below. Notice that the Max Free block found was equal to a quarter of the Cache size before the fragmentation and then was reduced to a low value after the first flush.



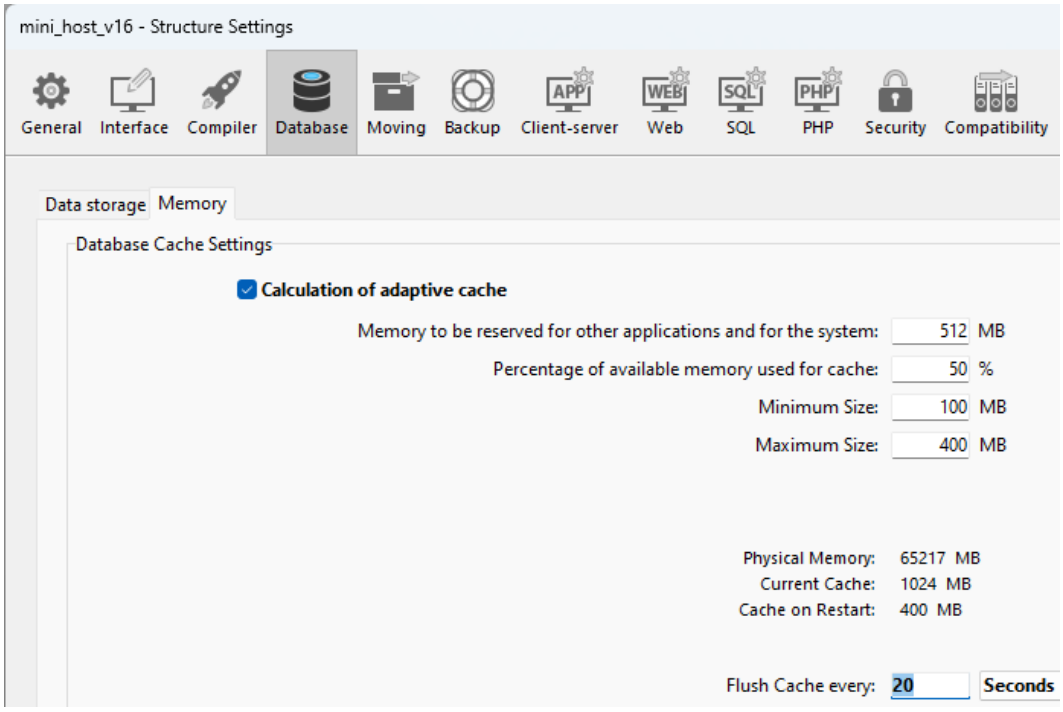


Note: In this historical graph from a Load testing done with 4D Server 13.2 HF1 (64-bit) via 4D_Info_Report, notice the regular flushing of the Cache by one quarter (dark green polygon versus light green line) and the related reduction of the Max free block.

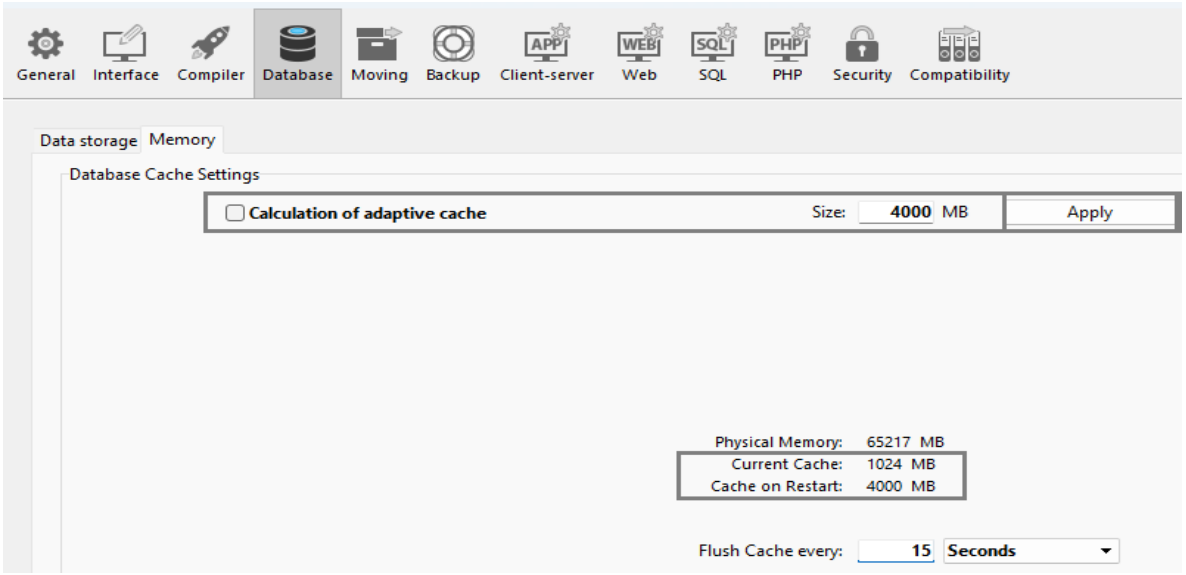
Starting with 4D Server v16

A Cache size change (and a flush Frequency change) can be applied without restarting the Server, either via the SET CACHE SIZE command or, more easily, through the interface.

The change is applied through the Structure Settings/Database/(Memory) tab:



If the Calculation of adaptative cache is unchecked, an “Apply” button is displayed to validate the change in the Cache size value.



Cache Storage vs. Memory Storage

What is moved to and stored in the Cache?

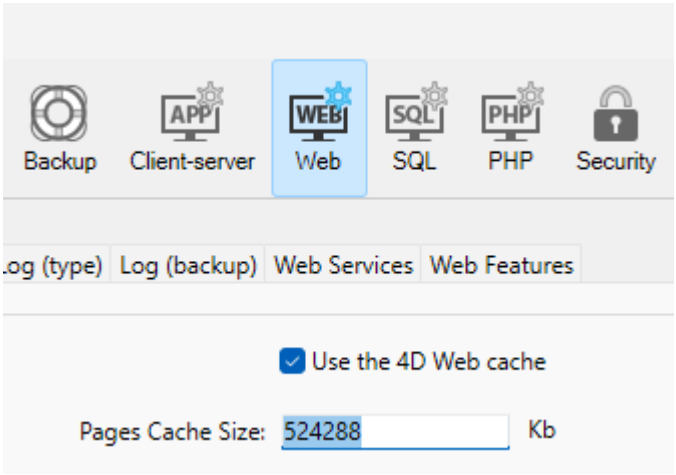
The list of the elements below represents what is stored in the cache:

- Structural definitions of tables, fields, relations, indexes, etc.
- General information about the open database (file paths, properties, etc.)
- Data file allocation bit tables
- Address tables for records, indexes, blobs, extra properties, etc.
- Index pages
- Records
- BLOBs (may be allocated in main memory instead if not enough space in Cache)
- Extra properties
- Sequence numbers
- Transactions
- Selections
- Sets
- Temporary buffers for sorting, read ahead, buffered disk write, etc.
- Structure objects (methods, forms, etc.)

4D constantly manages memory, juggling different kinds of objects to keep internal access as efficient as possible.

What is NOT stored in the Cache, but stored in memory?

- The 4D Web Server Cache if activated (Settings/Web/Options(I)):

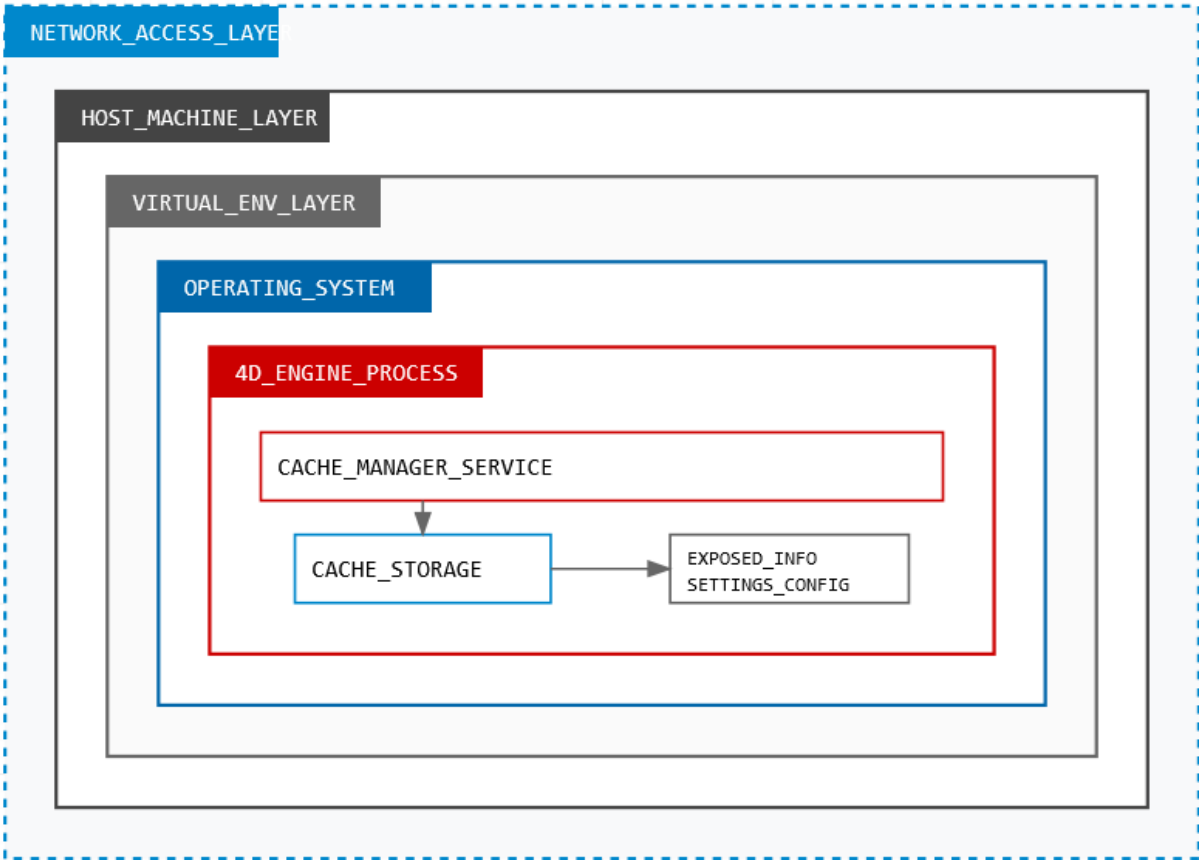


- All kinds of Storage content
- All activated Logs
- All activated operating system cache, such as opened files

How Does 4D Manage the Cache?

The question can be answered as follows: “It is a complex, private, and secured topic for 4D, the result of many years of improvement and adaptation…”.

For any user, with the secured credentials to login, consider the many layers to access to the cache: only the Settings of the Cache, and some Commands are available to users.



The organized content of the cache is not exposed:

- For obvious reasons of confidentiality regarding sensitive information.
- For future changes and optimizations.

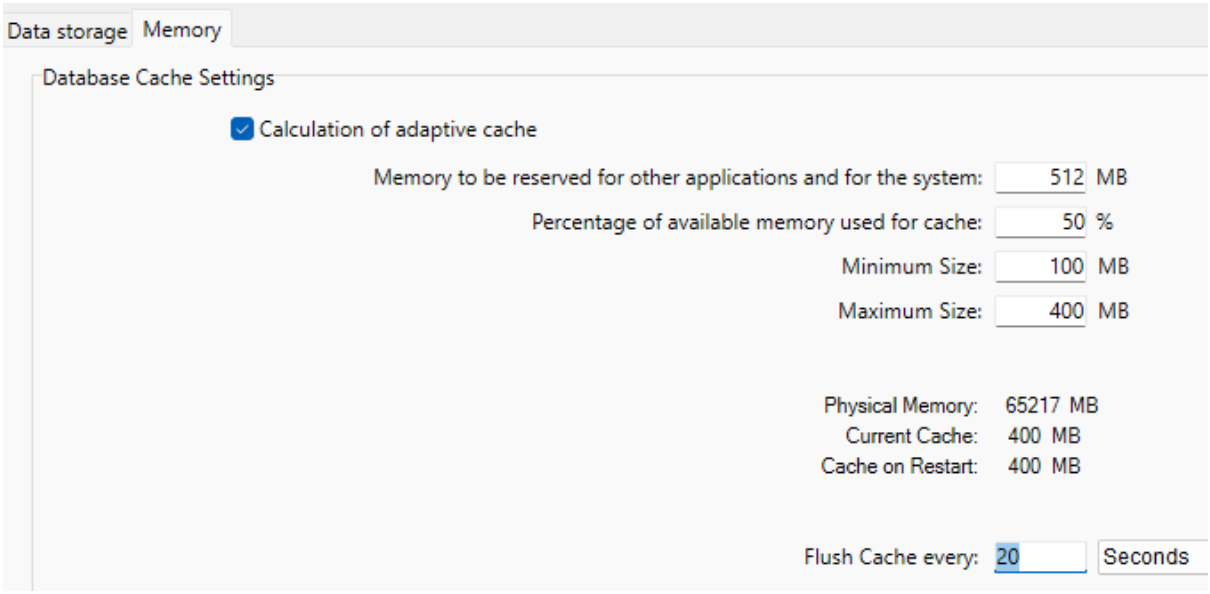
What can be retained:

At startup, even if no data was queried or created, the cache is loading certain elements for later data access (and structure/code access).

4D/4D Server will adapt to the current size of the cache and the available free memory, but it is up to the designer or administrator to verify that the current size is suitable for the database's usage.

By default, in a new project, the following settings are applied (adapted for Design mode):

- Calculation of adaptive cache is checked



Be aware that the Maximum Size is set to 400 MB, despite the 50 % percentage of available memory used for the cache, even with a large amount of Ram available.

Don't forget to enlarge this Maximum Size value when the data files are growing, or the database is deployed.

Note that the maximum size is set to 400 MB, despite 50% of the available memory being used for the cache, even when a significant amount of RAM is available.

Remember to increase this maximum size value as the data files grow or when the database is deployed.

If 4D needs to create some space in the Cache for new objects, it must first purge.

Key considerations:

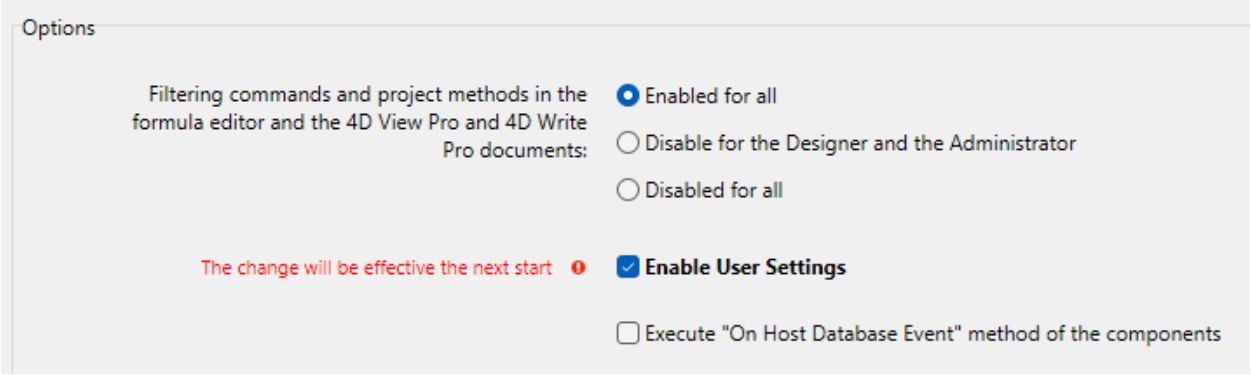
- 'ACID' compliance of 4D: ACID (atomicity, consistency, isolation, durability) refers to a set of properties for database transactions, designed to ensure data validity despite errors, power failures, and other mishaps. In databases, a sequence of operations meeting the ACID properties — often perceived as a single logical operation on the data — is referred to as a transaction. Transactions are often composed of multiple statements. Atomicity guarantees that each transaction is treated as a single "unit", which either succeeds completely or fails completely: if any of the statements constituting a transaction fails to complete, the entire transaction fails, and the database is left unchanged. An atomic system must guarantee atomicity in each situation, including power failures, errors, and crashes.
- 'CRUD': Create, Read, Update, and Delete: As data grows and ages, the frequency of Create, Update, and Delete operations will decrease relative to Read operations. To simulate cache usage during deployment, it is important to consider the evolution of CRUD operations when creating a large dataset for load testing, as this better reflects the general usage patterns of a large database.

The Cache functions like a dynamic resource: if the server is not restarted, it will continuously store the most frequently accessed content for connected users and be more responsive to the most frequent requests.

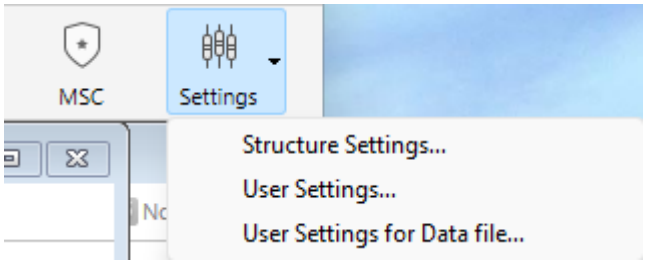
The settings files including the Cache for deployment

When some Cache Settings are applied for a Database (Structure Settings / Database/Memory), the settings are applied locally by default, and are set for the computer, adapted first for design testing and local load testing.

If the database must be deployed with separate settings for different existing data files, one setting in the preference must be checked (Settings/Security/User Settings):



If the “User Settings” is checked, when restarting the database, this new popup is displayed for the settings:



If a change is applied in the Settings for User Settings or User Settings for Data file, a new json file “settings.4DSettings” is added or updated in a “Settings” folder:

For example, changing some settings for the Cache:

```
<?xml version="1.0" encoding="UTF-8"?><preferences stamp="2">
    <com.4d>
        <database>
            <cache flush_buffer_delay="20s" size_maximum="4000"/>
        </database>
    </com.4d>
</preferences>
```

- User Settings: File located in the Database folder (inside the “Settings” folder)
- User Settings for Data file: File next to Data file (inside the “Settings” folder)

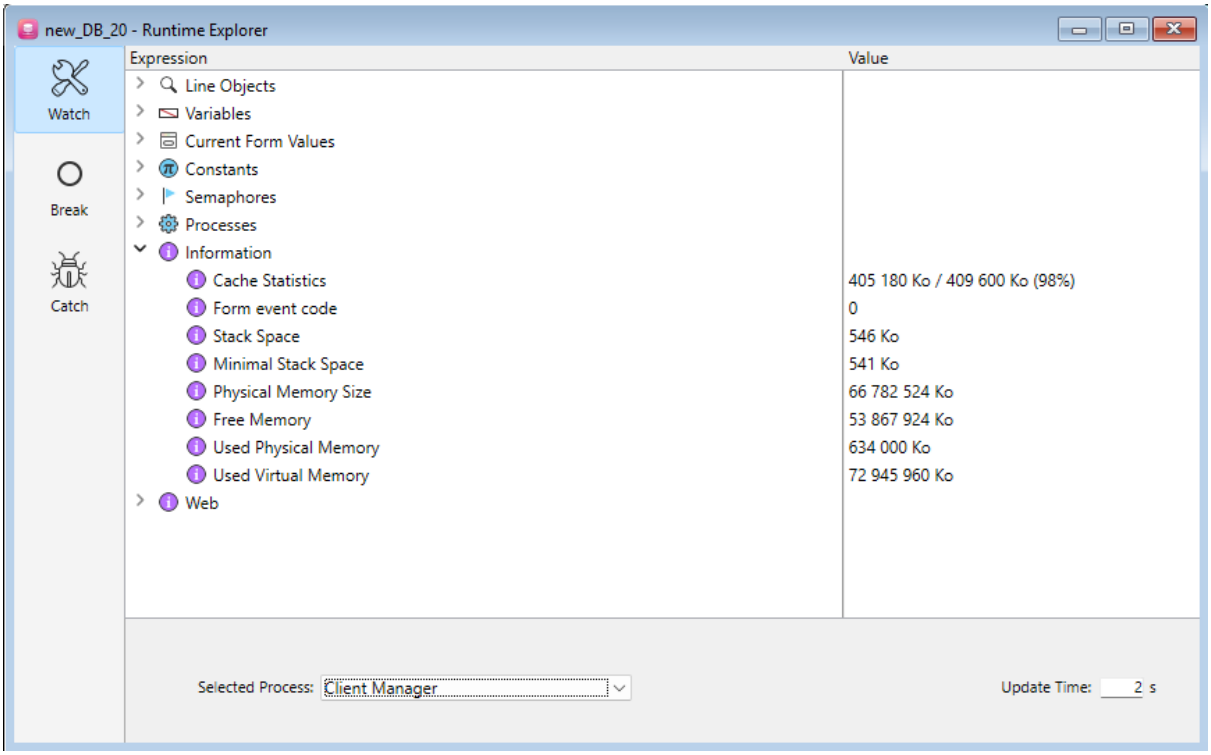
For more information on configuring cache settings, refer to the documentation:
https://developer.4d.com/docs/20/Desktop/user-settings

Visualizing the cache values through the interface

The run time explorer and the administration windows are available if the 4D/4D Server is not run as service or executed in headless mode (introduced in 4D 18: https://blog.4d.com/headless-4d-applications/).

Cache information can be viewed via several 4D server windows:

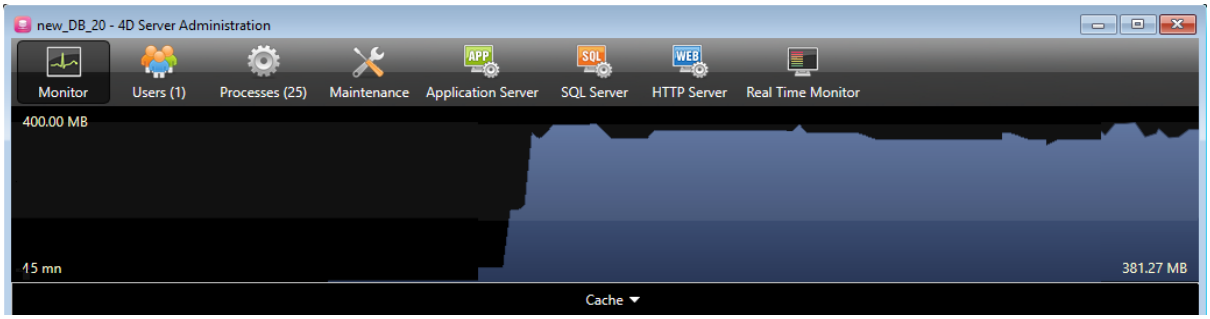
- Via the Runtime Explorer (when available), in “Information”:



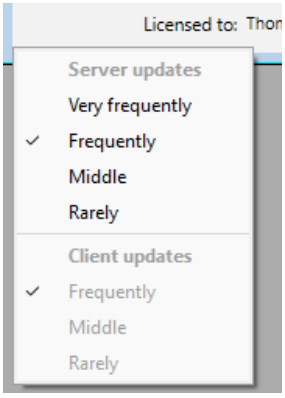
Note: In ‘Information’ are exposed the values of the GET MEMORY STATISTICS command

- Via the interface of the Administration window, the Cache usage history is visible.

For more information, refer to the documentation:
https://developer.4d.com/docs/20/ServerWindow/monitor



Note: A hidden popup window available via a right-click in the lower left corner of the Administration window let the user change the frequency of the refresh (Frequently is set by default)

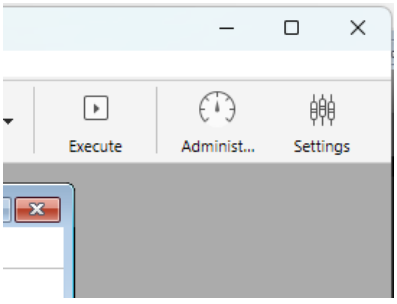


The recommendation to avoid a rather useless CPU load of the 4D Server is to close the Administration window if the observation of the current values is not needed: if the window is opened again, selecting the Monitor/Cache will display the updated historical Cache usage again (or at least reduce the Server updates frequency via the popup).

Reminder: When 4D Server is quitted, it reminds if the “Administration window” was closed or not, the window will open again or not by default next time the server is started.

- Opening Administration window from Remote Machines

Via a 4D Remote, in Design mode, click on the “Administration” icon to open the server administration window:




```
If (Application type=4D Remote mode) // display via the language
    // Do not provide this access to unauthorized remove users!
    OPEN ADMINISTRATION WINDOW
End if
```

Note: To open a server administration window from a remote machine, the user must be connected to the remote database as a Designer or Administrator. Otherwise, attempting to open the administration window generates a privilege error (-9991).

Even if the Server has been started as a Service, or in headless mode, the Administration window is available to authorized remote users.

Note: When the Administration window is displayed on 4D Remote, the Cache information does not replicate the graphic historical information available on 4D Server

Retrieving Cache values through 4D commands

Via the use of **GET MEMORY STATISTICS**

GET MEMORY STATISTICS (renamed from **GET CACHE STATISTICS** in 4D v13, renamed **MEMORY STATISTICS** starting with 20 R7).

This command provides information about the memory usage and the Cache usage of 4D/4D Server. The real values

GET MEMORY STATISTICS (info type ; arrNames ; arrValues ; arrCount)

infoType	Longint	input	addition of constants to specify the type of info desired
arrNames	Text array	output	name of the kind of value
arrValues	Real array	output	info values
arrCount	Real array	output	count of objects for the info considered (when available)

The infoType parameter is 1 (only supported value on recent versions of 4D)

1 – General memory info such as that displayed in the Runtime Explorer (physical, virtual, free, and used memory space, etc.)

Example of getting the available info:

```
GET MEMORY STATISTICS(1 ; arrNames ; arrValues ; arrCount) // MEMORY STATISTICS 20R7
```

This will return the total size of the Cache, the used size of the Cache, the number of blocks in the Cache, etc.:

Value in array names	Description of the array values
cacheSize	Maximum Cache size for the database. Note this is the actual Cache that was allocated and might not match the value you set in the settings.
usedCacheSize	Amount of currently used Cache.
Physical Memory Size	Total RAM on the machine.
Free Memory	Free RAM on the machine (as reported by the OS).
Used physical memory	Amount of RAM used by 4D (reported as Working Set on Windows, Real Memory on Mac OS X)
Used virtual memory	(The returned value might look huge starting with 4D 19R5, as much more virtual memory is reserved)
Stack memory	Total of the stack size of all process
Free stack memory	(can have the same total value as Stack memory)

Via the command Get database measures (created in 4D v14.3)

The **Get database measures** command returns detailed information about 4D database engine events. The information includes data read/write access from/to the disk or the memory cache, as well as the use of database indexes, queries and sorts.

Get database measures returns a single object containing all the relevant measures. The options object parameter allows to set options for the returned information.

Overview of the returned object: The returned object contains a single property named "DB" that has the following basic structure:

```
{
   "DB": {
      "diskReadBytes": {...},
      "cacheReadBytes": {...},
      "cacheMissBytes": {...},
      "diskWriteBytes": {...},

      "diskReadCount": {...},
      "cacheReadCount": {...},
      "cacheMissCount": {...},
```

```
        "diskWriteCount": {...},

        "dataSegment1": {...},
        "indexSegment": {...},

        "tables": {...},
        "indexes": {...}

    }
}
```

Elementary properties

Elementary properties can be found at different levels in the DB object. The properties return the same information but at different scopes. Below a description of the elementary properties:

Name	Information returned
diskReadBytes	Bytes read from disk
cacheReadBytes	Bytes read from cache
cacheMissBytes	Bytes missed from cache
diskWriteBytes	Bytes written to disk
diskReadCount	Read accesses from disk
cacheReadCount	Read accesses from cache
cacheMissCount	Read accesses missed from cache
diskWriteCount	Write accesses to disk

Via the command Cache info (created in 4D v16)

The **Cache info** command returns an object that contains detailed information about the current cache contents (used memory, loaded tables and indexes, etc.)

Note: This command only works in local mode (4D Server and 4D); it must not be used from 4D in remote mode.

By default, returned information refers to the running database only. The optional dbFilter object parameter allows to specify the command scope:

- Pass the "dbFilter" attribute with the "All" value to get cache information about all running databases, including components.
- Pass the "dbFilter" attribute with a "" (empty string) value to get information about the current database only (equivalent to omitting the dbFilter parameter).

The **Cache info** command returns a single object containing all the relevant information about the cache. The returned object has the following basic structure:

```
{
   "maxMem": Maximum cache size (real),
```

```
    "usedMem": Current cache size (real),
    "objects": [...] Array of objects currently loaded in cache
}
```

$info: Returns information about the entity sets currently stored in 4D Server's cache as well as user sessions. Check the Rest documentation:
https://developer.4d.com/docs/20/REST/info

Description

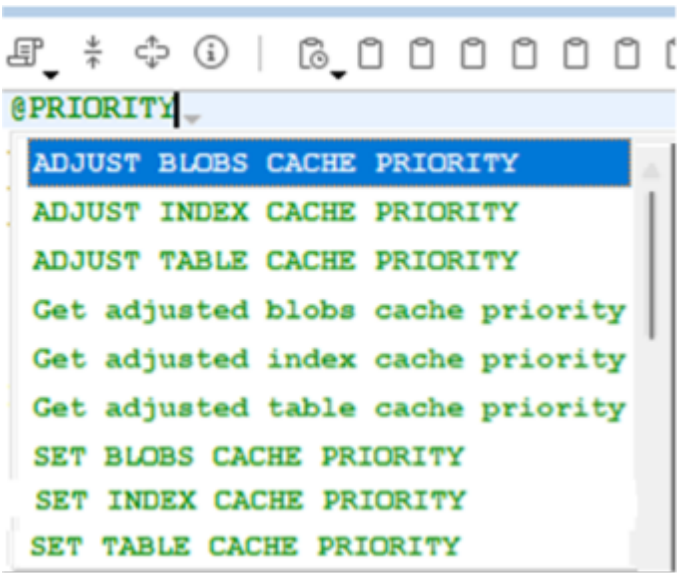
If the request is called in a project, information in the following properties is retrieved:

Property	Type	Description
cacheSize	Number	4D Server's cache size.
usedCache	Number	How much of 4D Server's cache has been used.

Note: The returned object can be quite huge if many process are running on the 4D Server. Better use this command for tuning or support reasons.

Via the command Priority in Cache (Set and Get)

The list of available commands is reminded, but unless for fine tuning after test, there is no obvious reason to use them, as the Cache will automatically adjust the content for the most demanded content. For more details, check the Documentation:
https://developer.4d.com/docs/category/cache-management



DataStoreClass information

A Datastore is the interface object provided by ORDA to reference and access a database. Datastore objects are returned by the following commands documentation: https://developer.4d.com/docs/20/API/DataStoreClass#flushandlock:

ds: a shortcut to the main datastore

Open datastore: to open any remote datastore

Function.**flushAndLock**(): The .**flushAndLock**() function flushes the cache of the local datastore and prevents other processes from performing write operations on the database. The datastore is set to a consistent, frozen state. Calling this function is necessary before executing an application snapshot, for example.

Monitoring the cache via 4D_info_report

The 4D_Info_Report component has been updated for years and is maintained. It is compatible with the latest current and future versions of 4D. The component creates human readable text reports as a snapshot (single report) of the state of the Server, or a series of reports via a stored procedure for historical and evolution analysis.

- Advantage of using this component: even if the Server is running as Service or in headless mode, the remote (Administrator) user can check in real time the evolution of the running database and later adjust for example the cache settings based on the displayed polygons or illustrate a memory leak.

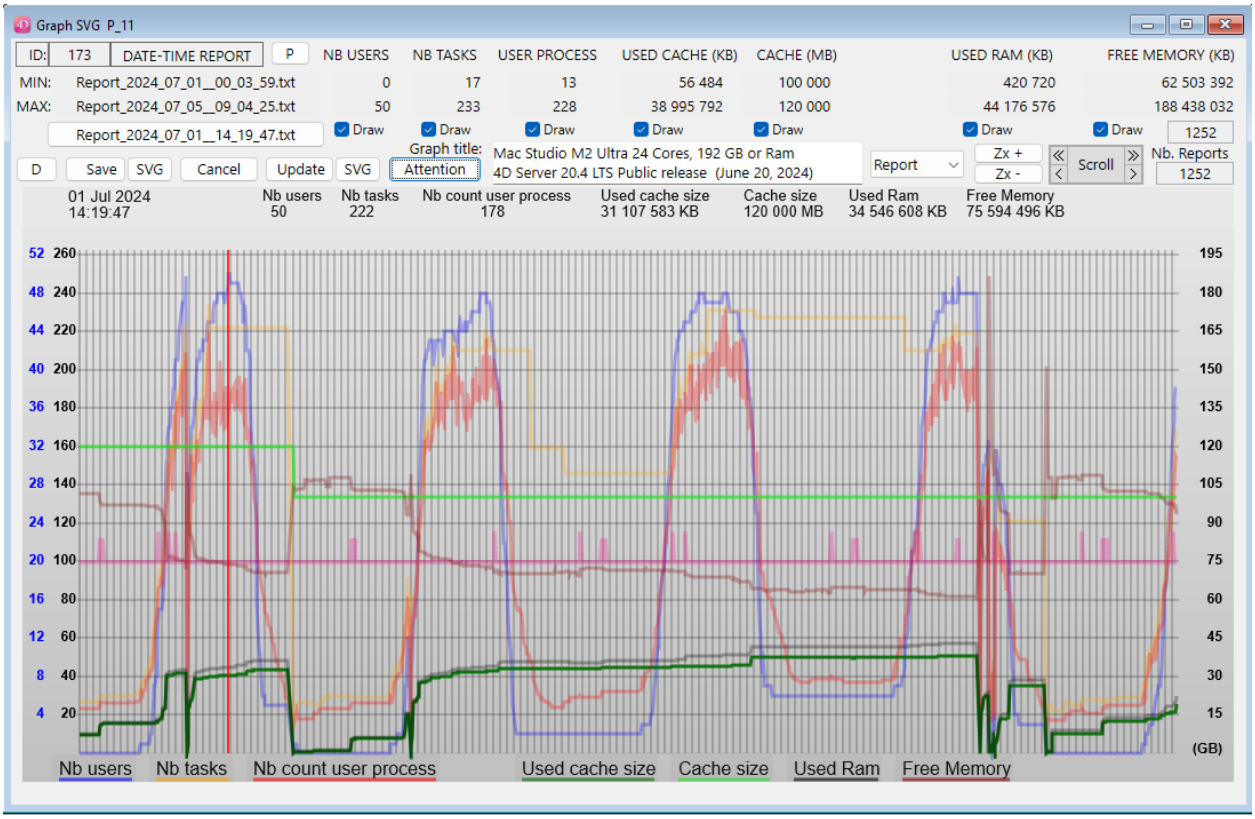
An archive of the Folder_reports (default name of the folder where the created reports are stored) can be provided to another staff or a technical support member for analysis.

A single report (human readable) provides much information about the context of execution of 4D or 4D Server.

The component is available via Github: https://github.com/4d/4D_Info_Report/releases/latest/.

(An archive of components for older versions of 4D are available via the GitHub link).

A basic graphic interface is embedded in the component, allowing the view of the evolution of some values logged on the Server via the stored reports, and some Information/Attention content. for example, on the remote application, when the Server is running, and component is implemented.



The two represented values about the Cache are:

- The light green polygon (flat unless a change is applied to the Cache size)
- The dark green polygon representing the filling level of the Cache.

Both green polygons are scaled (like other values about memory) on the right scale, expressed in MB or GB.

Note: For an in-depth understanding of the component, please check this Technote: 4D_Info_Report Tips and Support Cases (February 2024):https://kb.4d.com/assetid=79390

Other Cache Types: ORDA cache

4D remote applications using ORDA to access the main datastore with the ds command. Note that the 4D remote application can still access the database in classic mode. The accesses are handled by the 4D application server.

Other 4D applications (4D remote, 4D Server) opening a session on the remote datastore through the Open datastore command. The accesses are handled by the HTTP REST server. Documentation: https://developer.4d.com/docs/20/ORDA/datastores/

A limited remote Cache (on the remote computer) has been introduced when using ORDA: Documentation: https://developer.4d.com/docs/20/ORDA/datastores#orda-cache

ORDA cache

For optimization reasons, data requested from the server via ORDA is loaded in the ORDA remote cache (which is different from the 4D cache). The ORDA cache is organized by dataclass, and expires after 30 seconds.

The data contained in the cache is considered as expired when the timeout is reached. Any access to expired data will send a request to the server. Expired data remains in the cache until space is needed.

By default, the ORDA cache is transparently handled by 4D. However, the contents can be controled using the following ORDA class functions:

- dataClass.setRemoteCacheSettings()
- dataClass.getRemoteCache()
- dataClass.clearRemoteCache()

Conclusion

The 4D Cache is an important part of 4D Server performance. Except other parameters that impact the Cache usage performance, like hardware configuration (available memory on the computer, storage kind, network access, Database design) This Technical Note is an update of the possible interactions with the Cache settings and observations, and what has changed about the Cache management up to the recent versions of 4D.