Server Memory Allocation White Paper

VERSION 2020

jade

Contents

Server Memory Allocation

The primary caching paradigm for database blocks changed in JADE 7.0, with the provision of tuning parameters that you should consider using to maximize performance.

This document outlines the changes made and the steps to take to tune the new cache. Several different scenarios are discussed as examples of how to tune for different situations.

For more guidance in improving the performance of a JADE system, see the *Performance Analysis* and *JADE Performance Design Tips* white papers, at https://www.jadeworld.com/jade-platform/developer-centre/documentation/white-papers.

Note This white paper is relevant to JADE release 7.0 and higher, including JADE 2020. In earlier releases, a number of mechanisms operated differently, and it is likely that many may also change with future releases. The comparison with JADE 6.3 is still useful because many systems were initially configured for JADE 6.3 or earlier, and have not been updated since.

Any description of non-JADE tools is thought to be accurate at the time of publication. Their operation may change at any time. Consult their documentation for up-to-date information.

Overview

Starting with JADE 7.0, the Persistent Database (PDB) uses its own disk cache as the primary cache for database objects, while JADE 6.3 and earlier used Windows file-system cache. This new cache allows for more flexibility and performance optimization than was available with Windows file-system cache.

Every JADE 7 database has a PDB disk cache in the database server. All JADE nodes and server applications contend for this disk cache. All objects that are read in from the database on disk are read through this cache. All persistent objects that are created or updated are written through disk cache. The PDB module uses direct I/O to read and write database blocks.

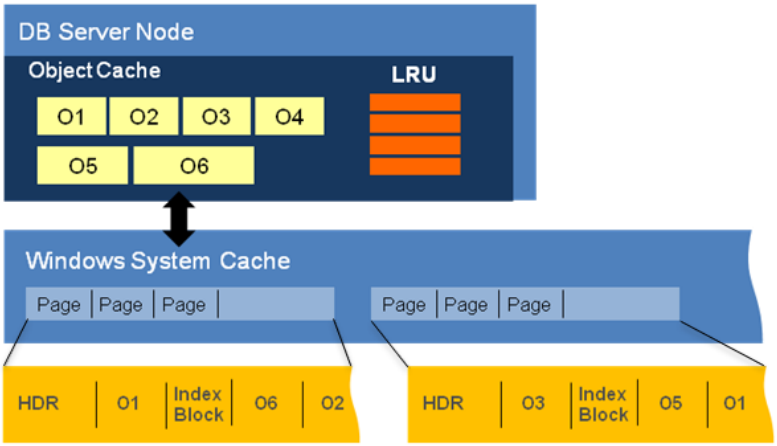
Note The PDB disk cache is separate from the server node JADE Object Manager (JOM) persistent object cache. The disk cache is directly used only by the PDB itself, to store images of database blocks from disk. The server node JOM persistent object cache stores individual objects, and is shared by server applications and **serverExecution** methods, similar to the way the JOM persistent object cache in any node is shared by all processes running in that node. For more details about the JADE caches, see the *Performance Analysis* white paper, at https://www.jadeworld.com/jade-platform/developer-centre/documentation/white-papers.

Essential Concepts

This section contains the following topics.

JADE 6.3 Database Cache Architecture

The database cache architecture in JADE release 6.3 and earlier is shown in the following diagram.

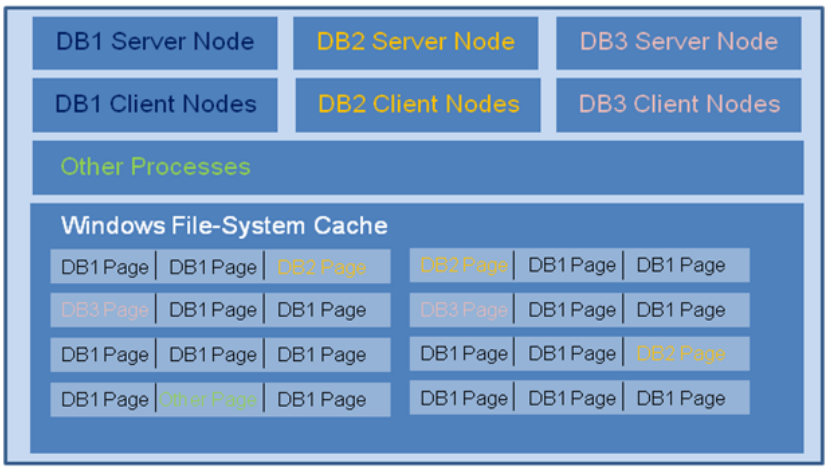


In JADE 6.3 and earlier, the database server maintained a small local object cache, whose size was specified in the **ObjectCacheSizeLimit** parameter in the [PersistentDb] section of the JADE initialization file. Windows maintains a cache of data read from files in its file-system cache, using all memory that is not otherwise in use. On a properly-configured production machine hosting JADE databases, Windows file-system cache normally occupied more than 50 percent of physical memory.

Windows maintained this file-system cache for all running processes, and all of the processes shared the same cache.

JADE 6.3 Physical Memory Layout

The physical memory layout in JADE release 6.3 and earlier is shown in the following image.

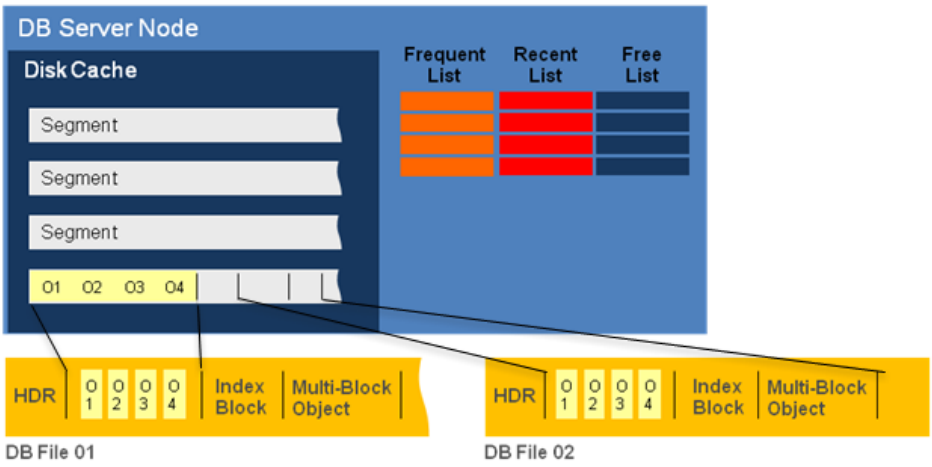


Note The Windows file-system cache has pages from the database files of **DB1**, **DB2**, and **DB3**, and from other processes as well. Windows allocated the available physical memory based on the data that is most-recently accessed, regardless of which database or process accesses it.

In the example in the previous image, most of the pages in Windows file-system cache are from **DB1**, because that database recently was the most active.

JADE 7 Database Cache Architecture

The database cache architecture in JADE release 7.0 and later is shown in the following diagram.



The JADE 7 database server has no small local object cache, instead maintaining its own cache of data read in from files. The size of this disk cache is specified with parameters in the [PersistentDb] section of the JADE initialization file; for example, the **DiskCacheMinSegments** and **DiskCacheMaxSegments** parameters.

Note Each JADE database server maintains its own disk cache; Windows file-system cache is not used for the database files. As there is no external manager to distribute memory amongst JADE databases, each database needs to be allocated an appropriate amount of memory to use.

Cache-Warming Mode

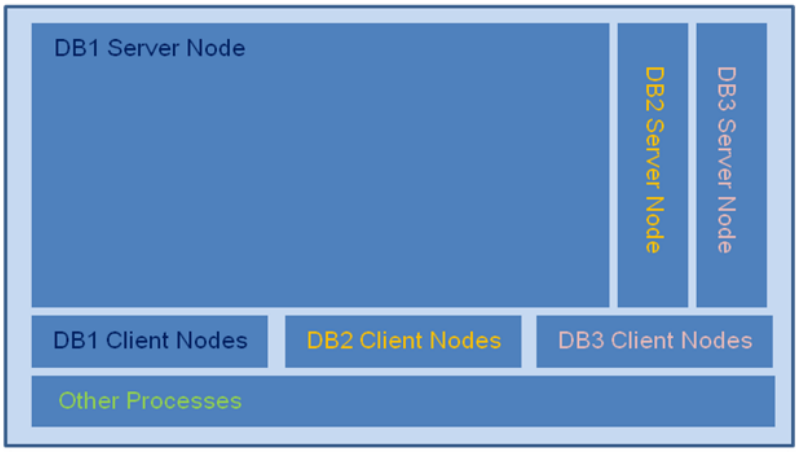
The database consists of a number of 8K byte blocks. A database block can contain multiple objects; alternatively, a single object can span multiple blocks.

When the database server starts up and has not yet reached the value of the **DiskCacheMaxSegments** parameter, it is in cache-warming mode. When a block is requested in cache-warming mode, eight additional blocks are read in from disk, with the intention being to accelerate the population of disk cache during database start-up. The blocks that were originally requested are read in and waited for, while the additional blocks are read asynchronously.

After the value of the **DiskCacheMaxSegments** parameter has been reached, individual blocks are read in, replacing other blocks in the cache that have not been used for a while.

JADE 7.0 Physical Memory Layout

The physical memory layout in JADE release 7.0 and later is shown in the following image.



In the example in this image, a system administrator has set the memory usage limits for each database, based on knowledge of the databases and the frequency of access.

Analyzing and Tuning Disk Cache

This section contains the following topics.

The Purpose of Cache

The main purpose of cache is to hold data that is likely to be used again in memory, therefore avoiding having to read the data in from disk again. If the cache is too:

- Small, the same database blocks are repeatedly read in from disk.
- Large, memory is wasted by holding blocks that are never re-used.

The fundamental goal of tuning the caches is to minimize the number of database blocks that need to be re-read from disk, for all databases on the machine.

The DiskCacheMaxSegments Parameter

The **DiskCacheMaxSegments** parameter in the [PersistentDb] section of the JADE initialization file is the most-important disk cache tuning parameter. It allocates the maximum size of the persistent database's disk cache, and therefore that database's share of the machine's overall physical memory.

Keep in mind that the **DiskCacheMaxSegments** parameter only specifies the size of the disk cache; not the total memory usage of the database server program (that is, **jadrap** or **jadserv**). The total memory usage is effectively the size of the disk cache plus the size of the database server program in JADE release 6.3 and earlier.

Initial Settings of the DiskCacheMaxSegments Parameter

As a starting point for any database, you could set the value of the **DiskCacheMaxSegments** parameter to a very large value, see how large it grows with 24 hours of normal activity, and then set the value of the parameter to two-thirds of that value, for example.

You could then adjust this up or down, depending on whether blocks are likely to be reused.

The DiskCacheMinSegments Parameter

The **DiskCacheMinSegments** parameter in the [PersistentDb] section of the JADE initialization file specifies what gets pre-allocated when the database server starts up. If a number of segments are quickly added each time the database server starts up, you could decide to set the value of the **DiskCacheMinSegments** parameter to something close to that number, so that they all are allocated at once.

Note, however, that the disk cache will never shrink below the value of the **DiskCacheMinSegments** parameter, even if it means that the **DiskCacheFreeMemoryTarget** parameter value has to be ignored. In this case, you must be certain that there is enough memory, to avoid paging by Windows. If the values of the **DiskCacheMinSegments** and **DiskCacheMaxSegments** parameters are set to the same value, *all* segments are allocated on start-up.

Measuring Disk Cache Performance with JADE Monitor

Several database disk cache values can be measured using the JADE Monitor. Two of these metrics are particularly useful in finding the optimum size of a database's disk cache, as follows.

- **Buffer Reassigns**, which indicates the cache is cycling buffers, from a least-recently used list.

  This is normal for a cache that has assigned all of its segments; for example, when the **DiskCacheMaxSegments** parameter is set to 100 and it has assigned 100 segments.

- **Buffer Steals**, which usually indicates that disk cache is too small. When a steal occurs, a buffer must be flushed to disk, which is a much more-expensive operation than a buffer reassign. If a disk cache is suitably configured for the activity of a JADE system, this metric should stay very close to zero (**0**).

  Note, however, that some steals can occur during allocation of a new segment if the disk cache is flooded with changed buffers; for example, during a data load.

If an excessively large value of the **DiskCacheMaxSegments** parameter is specified, the disk cache continues to grow and no buffers are recycled. In this case, the disk cache will contain large amounts of unnecessary objects, which could be inefficient, especially if the memory could be better used by another process.

You should review these two metrics periodically, and adjust the value of the **DiskCacheMaxSegments** parameter accordingly. If buffer steals cannot be eliminated, an increase in physical memory could be advisable.

Physical Memory Allocation Strategies

This section contains the following topics, for JADE 7.0 and higher.

What Is Recommended

This section contains the following topics.

Production Server, Single Database

Assuming that there is little else other than the JADE database and associated JADE nodes running on the server, and that the database is larger than physical memory, set the value of the **DiskCacheMaxSegments** parameter in the [PersistentDb] section of the JADE initialization file to most of the available memory. Let it default initially, and then increase it when you know how much memory is left over during peak times. To find out how many segments are in use, look for lines in the **jommsg** log files like those in the following example.

```
2014/09/20 02:39:05.331 00718-0aa0 PDB: DiskCache: segment allocated: _
poolSegments= 16, pool size now 1073741824 bytes (131072 blocks)
```

Using the default value of the **DiskCacheBlocksPerSegment** parameter value, 16 segments is equivalent to 1G byte.

Tip You should consider leaving a small amount unallocated, to allow for the unexpected. For example, if the database is hundreds of gigabytes and the machine has 32G bytes total physical memory, you could allocate the database disk cache 24G bytes and leave 2 to 4G bytes free, depending on your other requirements.

Production Server, Multiple Databases

Allocate memory amongst the databases involved based on performance requirements and activity. For example, consider the following circumstance.

- **DB1** is your main production database, and **DB2** and **DB3** are other utility databases
- The performance of **DB1** is critical
- **DB2** and **DB3** access very little data during a day, or they are accessed infrequently and their performance is not important
- There is 32G bytes of total physical memory

A reasonable course of action would be to allocate 1 or 2G bytes (or less) of disk cache to **DB2** and **DB3**, and allocate the remainder to **DB1**. As with the single database scenario, plan to leave some memory free to allow for the unexpected.

Remember that the **DiskCacheMaxSegments** parameter only specifies the size of the disk cache; not the total memory consumption of the database server.

Development Server, Shared with Other Processes

Limit the development databases to a fairly small size, perhaps 1 or 2G bytes of memory, depending on usage, to keep them from consuming more memory than they need.

If you do not restrict them, they will continue to consume more and more memory as different data is accessed. This can result in very old data being kept in physical memory, while other processes are starved for memory. It can also result in disk cache being paged out to the Windows paging file, which can lead to very poor performance.

DiskCacheBlocksPerSegment Parameter

The **DiskCacheBlocksPerSegment** parameter is not a performance-tuning parameter. It specifies the segment size used by the disk cache; that is, the unit of memory allocation when the cache has to grow.

The default size results in allocations and de-allocations of 64M byte chunks of address space. If a very large amount of physical memory is present, you could consider having fewer larger segments; for example, growing at 1G byte at a time.

DiskCacheFreeMemoryTarget Parameter

The **DiskCacheFreeMemoryTarget** parameter specifies the amount of physical memory that the disk cache tries to keep free while honoring pool minimum and maximum segment settings.

Ideally, the value of the **DiskCacheFreeMemoryTarget** parameter should never be reached for any database. This is achieved by ensuring that when all databases have reached the value of the **DiskCacheMaxSegments** parameter, there is still some "available memory" (as reported by the Windows Task Manager or the JADE Systems Manager).

Having said that, you could achieve greater security by setting the value of the **DiskCacheFreeMemoryTarget** parameter higher for low-priority databases. That way, if the unexpected *does* happen, the primary databases are less likely to be affected, and the low-priority databases act like the "canary in the coal mine".

Windows Paging File Size

In JADE release 6.3 and earlier, the bulk of the database cache was kept in Windows file-system cache, which is not backed by the pagefile.

From JADE 7.0 and later, the bulk of the database cache is kept in the database server's disk cache, which *is* backed by the pagefile, so the pagefile therefore needs to be large enough to hold it.

Tip The recommendation is to let Windows manage the pagefile size, unless you have a good reason not to do so.

What Not To Do

This section contains the following topics.

Don't Rely On the Default Setting of DiskCacheMaxSegments

The default setting of the **DiskCacheMaxSegments** parameter in the [PersistentDb] section is equivalent to half of the total physical memory. If you use the default setting, the database server will use more and more memory as data is accessed, up to half of physical memory.

If there is one database only and it is larger than physical memory, you would probably get better performance by using more memory than the default value provides.

If there are multiple databases on the server, they will all try to allocate half of physical memory, and conflicts will result that can degrade performance.

Don't Rely On DiskCacheFreeMemoryTarget to Distribute Memory

You could set the value of the **DiskCacheMaxSegments** parameter to a high value for multiple databases and rely on the **DiskCacheFreeMemoryTarget** parameter value so that each database will give back memory when overall memory is under pressure.

One problem with this is that de-allocating segments can be very expensive, especially if the segment to be deallocated contains data that has been updated but has not yet been written to disk.

Another problem is that memory may be given back by an active database, while other less-busy databases hold on to all of theirs. This can happen if all databases use the same **DiskCacheFreeMemoryTarget** parameter value and the busy database happens to be the first one to notice that free memory is reducing.

Another problem occurs if the database server is not given the permissions associated with the Windows Performance Monitor Users group. In this event, the amount of free memory cannot be determined, and over-allocation could result.

Note You should consider the **DiskCacheFreeMemoryTarget** parameter a safety net in case the unexpected happens; do not rely on using this setting to change allocations on the fly.

Over-allocation of Memory (Especially with Multiple Databases)

Remember that the default setting of the **DiskCacheMaxSegments** parameter is not suitable if you are running multiple databases on a single machine. Because it means that each database will use up to half of physical memory, it can result in database disk cache being paged out to the Windows paging file, which can lead to poor performance.

It is much better to restrict the memory consumption of each database server, to avoid the memory pressure that would cause Windows to page out the caches.

Even with a single database, specifying a **DiskCacheMaxSegments** parameter value that is too high can result in paging due to memory pressure.

Additional Windows Configuration Requirement

This section contains the following topic.

- Windows Performance Monitor Users Group

Windows Performance Monitor Users Group

In JADE release 7.0 or later, the Windows user running the **jadrap** program (or **jadserv**) needs Windows permissions to read the performance counters. This is achieved by being part of the Windows Performance Monitor Users group or having equivalent permissions.

If this is not the case, the **jommsg.log** will record messages similar to the following example and the JADE database will be unable to honor the value of the **DiskCacheFreeMemoryTarget** parameter.

```
2014/10/20 15:14:39.559 01c74-1c98 PDB: DiskCache Worker: PdhCollectQueryData for
Memory performance counters failed with PDH_NO_DATA result (0x800007d5)

2014/10/20 15:14:40.371 01c74-1c98 PDB: DiskCache Worker: Memory performance
counters can no longer be queried

2014/10/20 15:14:45.799 01c74-1c98 PDB: DiskCache Worker: Please add the
'MyDB1User' account into the 'Performance Monitor Users' group
```

For more details, see http://technet.microsoft.com/en-nz/library/cc749154.aspx.