How Much Memory Do I Need to Run Ingres?

Abstract

This article will attempt to explain how Ingres uses memory and how to calculate how much memory you need for a given configuration. It is based on Ingres running on Unix - although the principles are the same for Windows NT/2000 and VMS.

Ingres separates the memory it uses into logical areas called pools. In this document I will discuss the main memory pools which typically take up the most memory.

Whilst I have included information which may be useful for tuning these memory parameters – tuning is a separate and very large topic which I will touch on only briefly.

How Much Memory do I need to run Ingres?

If you want to skip the theory and jump straight to the calculations click here.

Static vs Dynamic Memory Pools

There are two types of pool - static and dynamic.

Static pools are allocated in full at start-up time. Static pools are allocated in shared memory with the exception of private DMF caches which are allocated in the DBMS process memory (see later).

Dynamic pools have a maximum size but no memory is allocated at start-up just an empty structure. Dynamic pools grow as required up to their maximum size. Dynamic pools are allocated in the DBMS process memory.

These are the static pools

- Logging system
- Locking system
- DMF Cache(s)
- Stack

These are the dynamic pools

- Optimizer Facility (OPF)
- Parser Facility (PSF)
- Query Execution Facility (QEF)
- Query Storage Facility (QSF)
- **Relation Description Facility (RDF)**

A summary of these pools is [here](#).

**Dynamic Pool Fragmentation**

Dynamic memory pools never shrink - they keep growing until they reach their maximum size. Since allocating memory is a relatively cpu-expensive operation Ingres keeps the memory segments it has allocated and tries to re-use them.

When a new memory segment is required Ingres first checks the existing ones for one at least as big as the one requested. If it finds one then this is used. If not it allocates a new segment of the requested size. If it cannot allocate a new one - because we have reached the maximum size of the pool - Ingres will try to de-allocate unused segments in order to free enough contiguous memory for the size of the requested segment.

Let's see that in a diagram. Here's my dynamic pool -

**After start-up it's empty**

![Diagram of a dynamic pool at start-up](image)

**A request for a segment comes along**

![Diagram of a dynamic pool after a request](image)
This segment is no longer needed (query commits)

A request for a bigger segment

Sometime later it might look like this

A request comes in for a large memory segment. None of the unused segments are big enough so Ingres does some re-allocation -
The effect of all this is that the dynamic memory pools become fragmented over time.

'Running out of' Memory

There are two sets of limits in place which could cause us to run out of memory.

The first set is the OS limits - there are limits on the number and size of shared memory segments available and limits on the size of memory a process can have.

The second set is the Ingres limits - the maximum sizes configured for the various memory pools.

Shared Memory OS Limits

Shared memory OS limits are usually set in the kernel configuration. Since the pools which go into shared memory are static we need to know that we can allocate the required amount in full before we start-up.

There is an Ingres program called syscheck which checks the shared memory limits (as well as semaphore and file descriptor limits). Syscheck is called by ingstart (or iistartup) and if the limits are too low Ingres will not start. You can run syscheck as a stand-alone utility - see this example output.

Process Memory OS Limits

Again the kernel configuration sets limits on how much memory a process can acquire. If Ingres tries to allocate more memory but is prevented due to an OS limit then an error will occur. Typically

\[ E_{SC0107\_BAD\_SIZE\_EXPAND} \text{ Error expanding virtual size of server.brk}() \]
failed with operating system error 12 (Not enough space)

This error will usually cause the DBMS to crash. To resolve either configure the DBMS to use less memory, or increase the memory available from the OS. Check the meaning of the OS error in the OS documentation - this should tell you which limit is being broken. (error 12 is the data segment size)

**Ingres Memory Limits**

If a request for a memory segment is refused because a dynamic pool has reached its limit (possibly due to fragmentation see above) then an error will occur and the SQL statement concerned will fail.

The exception to this is the DMF cache buffers and the log buffers. In these cases if all the buffers are in use the session does not fail but waits for a buffer to become available.

So it becomes clear that it is a very good idea to configure Ingres to always be within the OS limits since -

*Hitting an Ingres Limit causes the session to either have to wait or to fail*

*Hitting an OS limit may cause the DBMS to crash*

**What are these Memory Pools for?**

**Logging System**

The logging system memory largely consists of log buffers which are used to buffer reads/writes to the transaction logfile. It also includes some memory to keep track of active databases and transactions in the logfile.

**Locking System**

Each lock in Ingres is a memory structure. Memory is also allocated for hash tables which allow us to search quickly for particular locks.

**DMF Cache**

DMF Cache tuning is a whole topic by itself - this is just a few brief notes.

DMF Cache is the collection of buffers for reading/writing to database files.

Each DMF cache consists of a single-page cache and a group buffer cache. In Ingres II 2.0 and above there is a separate cache for each enabled page size.
An installation may have many caches depending on the DBMS configuration.

If a DBMS is configured with cache_sharing off (i.e. private cache) the cache memory is allocated in the process memory of that DBMS.

If the DBMS is configured with cache_sharing on then the cache is allocated in shared memory. There could be more than one shared cache and they are identified by cache name.

**Stack**

Stack space is allocated for each potential session in the DBMS. Some complicated queries use a large amount of stack. Queries with an IN clause with a large number of values can exhaust the stack.

**OPF**

This is memory used to optimize queries. The more complex the query the more memory is needed. Also creating large database procedures can often exhaust OPF memory.

There is a limit on how much OPF memory a single session can use. In 6.4 and OpenIngres this limit was controlled by opf_active – the presumed number of ‘active’ sessions using the OPF at any one time. In Ingres II a new parameter opf_maxmemf was introduced – this is the maximum amount of OPF memory a single session can use. It is expressed as a percentage of OPF memory. So the limits are -

<table>
<thead>
<tr>
<th>Ingres 6.4/OpenIngres limit</th>
<th>opf_memory/opf_active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingres II limit</td>
<td>The larger of (opf_memory/opf_active) and (opf_maxmemf/100)*opf_memory)</td>
</tr>
</tbody>
</table>

So it may be that if the overall opf_memory cannot be increased that more memory can be used by changing either opf_maxmemf or opf_active.

**PSF**

This is the memory used by the parser. Again complicated queries can sometimes exhaust the PSF memory.

There is also a per-session limit on PSF memory. Prior to Ingres II 2.5 this was controlled by the connect limit. In Ingres II 2.5 psf_maxmemf was introduced – the maximum amount of PSF memory a single session can obtain as a percentage of psf_memory. So the limits are -

<table>
<thead>
<tr>
<th>Pre II 2.5 limit</th>
<th>psf_memory/connect_limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>II 2.5 limit</td>
<td>The larger of (psf_memory/connect_limit) and (psf_maxmemf/100)*psf_memory)</td>
</tr>
</tbody>
</table>
QEF

This is the memory used by the Query Execution Facility. It contains the low-level instructions on how to perform the query together with some runtime environment information e.g. collation sequence, money and date formats. The amount of QEF memory allocated per session is based on qef_qep_mem – which is the presumed size of the low-level instructions for an average query. The amount allocated for query execution is

\[
2048 + (\text{connect\_limit} \times \text{cursor\_limit}) \times \text{qef\_qep\_mem}
\]

Another large component of QEF is the sort memory – qef_sort_mem. If a sort cannot be done in memory it will be done on disk in the work locations

QSF

The Query Storage Facility is where Query Execution Plans (QEPs) are stored. If you use repeated queries and/or large Database Procedures - you may want to increase qsf_memory.

RDF

Relation Description Facility is a cache for the table information - i.e. the structure, location, columns, datatypes etc. The more tables a query references the more rdf_memory is required.

Summary Table for the different Memory Pools

<table>
<thead>
<tr>
<th>Memory Pool</th>
<th>Type</th>
<th>Allocated in</th>
<th>Parameters affecting size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMF Cache (private)</td>
<td>Static</td>
<td>DBMS Process Memory</td>
<td>dmf_cache_size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmf_group_count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmf_group_size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmf_group_size (other page size equivalents)</td>
</tr>
<tr>
<td>DMF Cache (shared)</td>
<td>Static</td>
<td>Shared Memory</td>
<td>dmf_cache_size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmf_group_count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmf_group_size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmf_group_size (other page size equivalents)</td>
</tr>
<tr>
<td>Logging Segment</td>
<td>Static</td>
<td>Shared Memory</td>
<td>log.buffer_count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>log.database_limit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>log.tx_limit</td>
</tr>
<tr>
<td>Locking Segment</td>
<td>Static</td>
<td>Shared Memory</td>
<td>lock.hash_size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>lock.resource_hash</td>
</tr>
</tbody>
</table>
### Estimating Memory Requirements

Here's how to check how much memory your Ingres configuration will use. Remember you may use less than this if your dynamic pools never reach their limits before you restart Ingres.

All the shared memory requirements (logging, locking and any shared DMF caches) are checked by `syscheck` - so just run `syscheck`!

If you want to check the memory implications of a proposed change then you could back up `$II_SYSTEM/ingres/files/config.dat` and `protect.dat` - make the changes in CBF and then run `syscheck`. Afterwards revert to the original settings by restoring `config.dat` and `protect.dat`. You can do all this without restarting Ingres - so the changes never 'go live'.

The approximate size of a DBMS process is the sum of

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OPF</strong></td>
<td>Dynamic</td>
<td>Process Memory</td>
<td><code>opf_memory</code></td>
<td></td>
</tr>
<tr>
<td><strong>PSF</strong></td>
<td>Dynamic</td>
<td>Process Memory</td>
<td><code>psf_memory</code></td>
<td></td>
</tr>
<tr>
<td><strong>QEF</strong></td>
<td>Dynamic</td>
<td>Process Memory</td>
<td><code>qef_qep_mem</code></td>
<td><code>qef_sort_mem</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>connect_limit</code></td>
<td><code>cursor_limit</code></td>
</tr>
<tr>
<td><strong>QSF</strong></td>
<td>Dynamic</td>
<td>Process Memory</td>
<td><code>qsf_memory</code></td>
<td></td>
</tr>
<tr>
<td><strong>RDF</strong></td>
<td>Dynamic</td>
<td>Process Memory</td>
<td><code>rdf_memory</code></td>
<td></td>
</tr>
</tbody>
</table>

The approximate size of a DBMS process is the sum of

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OPF</strong></td>
<td></td>
<td></td>
<td><code>opf_memory</code></td>
<td></td>
</tr>
<tr>
<td><strong>QSF</strong></td>
<td></td>
<td></td>
<td><code>qsf_memory</code></td>
<td></td>
</tr>
<tr>
<td><strong>PSF</strong></td>
<td></td>
<td></td>
<td><code>psf_memory</code></td>
<td></td>
</tr>
<tr>
<td><strong>QEF</strong></td>
<td></td>
<td></td>
<td><code>qef_sort_mem</code></td>
<td><code>qef_qep_mem</code></td>
</tr>
<tr>
<td><strong>RDF</strong></td>
<td></td>
<td></td>
<td><code>rdf_memory</code></td>
<td></td>
</tr>
<tr>
<td><strong>Stack</strong></td>
<td></td>
<td></td>
<td>(stack_size*connect_limit)</td>
<td></td>
</tr>
<tr>
<td>Private DMF Cache (if present)</td>
<td>2k cache is turned 'ON' by default</td>
<td></td>
<td><code>dmf_memory</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>if cache.p4k_status is &quot;ON&quot;</td>
<td><code>p4k.dmf_memory</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>if cache.p8k_status is &quot;ON&quot;</td>
<td><code>p8k.dmf_memory</code></td>
</tr>
</tbody>
</table>
if cache.p16k_status is “ON”
    p16k.dmf_memory
if cache.p32k_status is “ON”
    p32k.dmf_memory
if cache.p64k_status is “ON”
    p64k.dmf_memory

Total

Approx. DBMS process size

DMF cache is the sum of the single page cache plus the group buffers –

\[
\text{dmf}\_\text{memory} = (\text{dmf}\_\text{cache}\_\text{size} \times \text{page}\_\text{size}) + \]

/* Single Page Cache */

\[
((\text{dmf}\_\text{group}\_\text{size} \times \text{dmf}\_\text{group}\_\text{count}) \times \text{page}\_\text{size}) \]

/* group buffers */

This calculation is given for reference – since dmf_memory is a derived parameter calculated automatically by cbf.

Remember that this is the potential size of a single DBMS server process of a given configuration. To get the total for the installation you need to work out the DBMS size for each DBMS configuration and multiply it by the startup_count for that DBMS type.

Don't forget to account for memory which may be in use by other processes on the server. Also if you configure too highly then your server may begin swapping/paging memory causing performance to suffer.

Checking memory (OS specific)

The following commands can be used to check memory sizes, allocation etc. These are from a server running Solaris. The commands may not be the same on other flavours of Unix - check your OS documentation.

Checking the size of DBMS server process

Solaris:

/usr/proc/bin/pmap 4722 (where 4722 is the pid of the iidbms process)

Check the total line at the end

total 32312K

This DBMS is using 32Mb of memory.

Alternate method -

ps -e -ovsz,args | grep iidbms
32312 /export/home/IngresII/ingres/bin/iidbms dbms (default) I

This DBMS is using 32Mb of memory.
Compaq Tru64:
ps -e -ovsize, args | grep iidbms

26.3M /export/maspa05/inges/bin/iidbms dbms (default) pm

Checking kernel process memory limits

ulimit -aH (as user ingres)
time(seconds) unlimited
file(blocks) unlimited
data(kbytes) unlimited
stack(kbytes) 261120
coredump(blocks) unlimited
nofiles(descriptors) 1024
vmemory(kbytes) unlimited

Note that the stack size here is not related to the DBMS parameter stack_size.

Checking for swapping

vmstat 2 5
procs 5 memory page disk faults
cpu r b w swap free re mf pi po fr de sr f0 s0 s2 -- in sy cs us
sy id
0 0 0 2944 320 0 1 0 0 0 0 0 0 0 0 0 28 96 47 0
1 99
0 0 0 107496 4872 0 7 0 0 0 0 0 0 0 0 0 27 89 51 0
0 99
0 0 0 107496 4872 0 7 0 0 0 0 0 0 0 0 0 29 232 61 3
3 94
0 0 0 107496 4872 0 0 0 0 0 0 0 0 0 0 33 93 55 0
1 99
0 0 0 107496 4872 0 2 0 0 0 0 0 0 0 0 0 28 122 52 0
2 98

The po column, 'pages out' (sometimes pout), should be 0. If not then paging/swapping to disk is occurring.

swapon -s
Total swap allocation:
Allocated space: 96685 pages (755MB)
Reserved space: 23357 pages (24%)
In-use space: 13852 pages (14%)
Available space: 73328 pages (75%)

This shows how swap is available/being used.
Example syscheck output

$ syscheck -v

Checking host "phaedrus" for system resources required to run Ingres...

46 file descriptors per-process required.
1024 is the current system hard limit.
1024 is the current system soft limit.

11665408 byte shared memory segment required by LG/LK sub-systems.
278528 byte shared memory segment required by DBMS server(s).
0 byte shared memory segment required by shared DBMS cache(s).
200000000 bytes is the maximum shared memory segment size.

3 shared memory segments required.
100 is the total number of shared memory segments allocated by the system.
100 shared memory segments are currently available.

35 semaphores required.
1000 is the total number of semaphores allocated by the system.
995 semaphores are currently available.

2 semaphore sets required.
100 is the total number of semaphore sets allocated by the system.

99 semaphore sets are currently available.

24869k bytes of swap space required.
173704k bytes is total swap space allocated by the system.
109248k bytes is the total swap space currently available.

Your system has sufficient resources to run Ingres.

$ syscheck

Checking host "phaedrus" for system resources required to run Ingres...

Your system has sufficient resources to run Ingres.