AIX shared memory behavior for 32-bit applications

Technote (FAQ)

Question

How does AIX handle shared memory for 32-bit applications, including IBM MQ C, C++ and COBOL server applications, and IBM MQ Java and JMS programs making bindings connections to local queue managers?

Cause

AIX handles shared memory for 64-bit programs much like other UNIX and Linux systems, but its rules for shared memory in 32-bit applications are unique. This document applies only to AIX 32-bit applications, including AIX 32-bit Java Virtual Machines.

Answer

AIX 32-bit Shared Memory

32-bit applications have a total address space of 4GB, and AIX divides this memory into sixteen equal segments of 256MB. The first hex character of any 32-bit address on AIX identifies its segment, for example address 0x7044A98C is in segment 0x7 (meaning 7), while 0xC41280A8 is in segment 0xC (meaning 12). Each of the sixteen memory segments is assigned one of three uses:

1. Native heap: Memory managed by the malloc(), calloc(), realloc() and free() native heap functions
2. Shared memory and mapped memory: Memory accessed using the shmat() or mmap() functions
3. Reserved by AIX: Some segments are reserved for use by the operating system

An AIX application can allocate many chunks of native heap memory from a single 256MB segment. An application can also mmap multiple chunks of memory in a single segment. However, when AIX attaches a shared memory set, it reserves an entire 256MB segment regardless of the size of the set. This is only true for 32-bit applications; AIX 64-bit applications only need enough space to fit the set.

Therefore, when a 32-bit application attaches a 10K shared memory set, AIX must reserve an entire 256MB segment, even though 255.99MB are unused. Uniquely, AIX allows 32-bit applications to use the shmctl() function to resize a shared memory set up to the 256MB segment size.

The maximum number of shared memory sets which a 32-bit AIX application can attach depends on how many segments in the address space are assigned for shared memory use. AIX has three different memory models which control the assignment of segments. The choice of which memory model to use is made with the -bD (also known as -bmaxdata) linker option when building the application:

Setting the memory model when building an application

```
aix> xlC_r -q32 -bD:0x40000000 MyApp1.c -o MyApp1 -I $MQ_INSTALLATION_PATH/inc -L $MQ_INSTALLATION_PATH/lib -lmqm_r

aix> xlC_r -q32 -bmaxdata:0x70000000 MyApp2.c -o MyApp2 -I $MQ_INSTALLATION_PATH/inc -L $MQ_INSTALLATION_PATH/lib -lmqm_r

aix> xlC_r -q32 -bD:0xC0000000/DSA MyApp3.c -o MyApp3 -I $MQ_INSTALLATION_PATH/inc -L $MQ_INSTALLATION_PATH/lib -lmqm_r
```

↑ Back to top

AIX 32-bit Memory Models

The default memory model applies when you don't use the -bD (or -bmaxdata) linker option while building an application, or when you use the value -bD:0x00000000. The default model restricts the native heap to segment 2, which means the application can allocate at most 256MB of native heap memory.

Applications using the default memory model can attach up to 11 shared memory sets, starting with segment 3 up through 12,

and finally using segment 14. If the application uses mmap, then fewer segments will be available for shared memory. The arrows below indicate the direction in which segment ranges are allocated:

Default Memory Model: 0x00000000

```
Segment 15          Reserved by AIX for shared library data
Segment 14          Shared memory / mmap
Segment 13          Reserved by AIX for shared library text
Segment  3 ▸ 12     Shared memory / mmap
Segment  2          Thread stacks and native heap
Segment  1          Reserved by AIX for program text
Segment  0          Reserved by AIX for kernel data
```

The large memory model applies when you build an application with the -bD:0xN0000000 option, where "N" is a number from 1 to 8 indicating the number of native heap segments. The large memory model starts the native heap in segment 3, growing up to segment N+2, so native heap sizes from 256MB to 2GB are possible. For example, if you choose N=6, the application is guaranteed 1.5GB of native heap space.

Shared memory and mmap start with the segment just above the native heap on up through 12, and finally using segment 14. Depending on the value of N the application can attach from 3 to 10 shared memory sets. In this example with N=6, up to 5 shared memory sets may be attached. If the application uses mmap, then fewer segments will be available for shared memory:

Large Memory Model, for example: 0x60000000

```
Segment 15          Reserved by AIX for shared library data
Segment 14          Shared memory / mmap
Segment 13          Reserved by AIX for shared library text
Segment  9 ▸ 12     Shared memory / mmap
Segment  3 ▸ 8      Native heap
Segment  2          Reserved by AIX for initial thread stack
Segment  1          Reserved by AIX for program text
Segment  0          Reserved by AIX for kernel data
```

The very large memory model applies when you build an application with the -bD:0xN0000000/DSA option, where "N" is a hexadecimal number from 0 to D (meaning 0 to 13) indicating the number of native heap segments. When "N" equals 0, B, C or D, (meaning 0, 11, 12 or 13) AIX removes its reservation on segments 13 and 15.

DSA stands for Dynamic Segment Allocation, which means that native heap memory is allocated from the lowest available

segment growing upwards, while shared memory sets are attached to the highest available segment first. Memory segments in the middle are available and will be used for either the native heap or for shared memory, depending on what the application needs first.

For example, using N=6 with DSA enabled, AIX will start the native heap in segment 3. The native heap may extend to segment 4 and up to segment 8, if those segments are still available when the application asks for more native memory. Meanwhile, shared memory and mmap start in segment 14, then go to segment 12 on down to 9 with no conflict. If more shared memory or mmap space is needed, then segments 8 down to 4 may be used if the native heap has not already used them.

Very Large Memory Model, for example: 0x60000000/DSA

Segment 15	Reserved by AIX for shared library data
Segment 14	Shared memory / mmap
Segment 13	Reserved by AIX for shared library text
Segment 12 ‣ 9	Shared memory / mmap
Segment 8 ‣ 4	Shared memory / mmap, if available
Segment 4 ‣ 8	Native heap, if available
Segment 3	Native heap
Segment 2	Initial thread stack
Segment 1	Reserved for AIX program text
Segment 0	Reserved for AIX kernel data

If you use N=B (meaning 11) with DSA enabled, AIX will start the native heap in segment 3. The native heap may extend to segment 4 and up to segment 13, if those segments are still available when the application asks for more native memory. Meanwhile, shared memory and mmap start in segment 15 and then use segment 14 with no conflict. If more shared memory or mmap space is needed, then segments 13 down to 4 may be used if the native heap has not already used them.

Very Large Memory Model, for example: 0xB0000000/DSA

Segment 15 ‣ 14	Shared memory / mmap
Segment 13 ‣ 4	Shared memory / mmap, if available
Segment 4 ‣ 13	Native heap, if available
Segment 3	Native heap
Segment 2	Initial thread stack
Segment 1	Reserved for AIX program text
Segment 0	Reserved for AIX kernel data

In order to change the memory model used by an application, you can rebuild the application and choose a different -bD (or -bmaxdata) option. You can also override the memory model of any application at runtime by setting the LDR_CNTRL environment variable before starting it. The value should be of the form MAXDATA=0xN00000000 or MAXDATA=0xN0000000@DSA. Note that the environment variable value is spelled @DSA even though the linker option is spelled /DSA:

Overriding the memory model at runtime

Use the env command to start a single application with a different memory model:

```
aix> env LDR_CNTRL=MAXDATA=0x30000000 MyApp1 -m QMGR1
```

Alternatively, export the variable to start multiple applications with a different memory model:

```
aix> export LDR_CNTRL=MAXDATA=0xA0000000@DSA
aix> MyApp2 -m QMGR2
aix> MyApp3 -m QMGR3
aix> unset LDR_CNTRL
```

↑ Back to top

The IBM 32-bit Java Virtual Machine

The IBM 32-bit Java Virtual Machine on AIX uses the very large memory model, but the exact value depends on the Java heap size as set by the -Xmx command line parameter. The IBM Java Virtual Machine uses native heap memory for some of its work, but all Java objects are allocated from an area called the Java heap. The Java heap is implemented using a contiguous piece of mmap memory created by the IBM Java Virtual Machine at startup. It the Java heap is not an even multiple of 256MB, then it will start midway through a segment, and the rest of the segment will be available.

For Java heap values up to and including 2.25GB (-Xmx2304M), the IBM Java Virtual Machine uses the very large memory model with maxdata set to 0xA0000000/DSA. If the Java heap is 256MB or less (-Xmx256M) then it will fit in segment 14. Otherwise, since AIX has segment 13 reserved, the Java heap will be found in segment 12 on down. In the example below with a 1.0GB Java heap, the Java heap uses segments 9 through 12:

Java Heaps up to 2.25GB, for example: -Xmx1024M

Segment 15	Reserved by AIX for shared library data
Segment 14	Shared memory / mmap
Segment 13	Reserved by AIX for shared library text
Segment 12	Java heap (mmap)
Segment 11	Java heap (mmap)
Segment 10	Java heap (mmap)
Segment 9	Java heap (mmap)
Segment 8 ‣ 4	Shared memory / mmap, if available
Segment 4 ‣ 8	Native heap, if available
Segment 3	Native heap
Segment 2	Initial thread stack
Segment 1	Reserved for AIX program text
Segment 0	Reserved for AIX kernel data

For Java heap values up to and including 3.0GB (-Xmx3072M), the IBM Java Virtual Machine uses the very large memory model with maxdata set to 0xB0000000/DSA. AIX removes its reservations on segments 13 and 15 at this point, leaving more contiguous memory for the Java heap. In the example below the Java heap size is 2.5GB:

Java Heaps up to 3.0GB, for example: -Xmx2560M

Segment 15	Java heap (mmap)
Segment 14	Java heap (mmap)
Segment 13	Java heap (mmap)
Segment 12	Java heap (mmap)
Segment 11	Java heap (mmap)
Segment 10	Java heap (mmap)
Segment 9	Java heap (mmap)
Segment 8	Java heap (mmap)
Segment 7	Java heap (mmap)

Segment	6	Java heap (mmap)
Segment	5 ▸ 4	Shared memory / mmap, if available
Segment	4 ▸ 5	Native heap, if available
Segment	3	Native heap
Segment	2	Initial thread stack
Segment	1	Reserved for AIX program text
Segment	0	Reserved for AIX kernel data

For Java heap values up to the maximum of 3.25GB (-Xmx3328M), the IBM Java Virtual Machine uses the very large memory model with maxdata set to 0x00000000/DSA, that is, one native heap segment in segment 2, similar to the default memory model. If you ask for the full 3.25GB for the Java heap there will be absolutely no room for any shared memory, but if you leave a little room then the Java heap will start midway through segment 3. In the example below, the Java heap is 3.125GB, leaving just 128MB of room for shared memory and other mmap activity:

Java Heaps up to 3.25GB, for example: `-Xmx3200M`

Segment	15	Java heap (mmap)
Segment	14	Java heap (mmap)
Segment	13	Java heap (mmap)
Segment	12	Java heap (mmap)
Segment	11	Java heap (mmap)
Segment	10	Java heap (mmap)
Segment	9	Java heap (mmap)
Segment	8	Java heap (mmap)
Segment	7	Java heap (mmap)
Segment	6	Java heap (mmap)
Segment	5	Java heap (mmap)
Segment	4	Java heap (mmap)
Segment	3	Java heap (mmap) and 128MB left for shared memory / mmap
Segment	2	Thread stacks and native heap
Segment	1	Reserved for AIX program text
Segment	0	Reserved for AIX kernel data

↑ Back to top

Using Extended Shared Memory with the EXTSHM Variable

AIX provides an extended shared memory capability that is enabled using an environment variable called EXTSHM. The AIX extended shared memory behavior abolishes some of the default rules and makes shared memory behave as it does on other UNIX and Linux systems. The EXTSHM environment variable has three values:

EXTSHM Values

1. **EXTSHM=ON** Shared memory sets less than 256 MB in size are not rounded up in size to a whole 256 MB address space segment. Internally, AIX uses mmap to implement shared memory when this variable is set.
2. **EXTSHM=1SEG** This is a synonym for EXTSHM=ON.
3. **EXTSHM=MSEG** All shared memory sets, regardless of size, are implemented using mmap.

When the EXTSHM variable is in use, shared memory sets are rounded up in size only to the nearest page (typically 4KB). This removes the otherwise tight restrictions on the number of shared memory sets which a 32-bit AIX application can attach. Now 32-bit applications can attach as many shared memory sets as they have room for.

Just like LDR_CNTRL, you can set the EXTSHM variable for individual applications or multiple applications. You can even add it to your login profile so that all applications you start use EXTSHM:

Setting EXTSHM

Use the env command to start a single application with extended shared memory enabled:

```
aix> env EXTSHM=ON MyApp1 -m QMGR1
```

Alternatively, export the variable to start multiple applications with extended shared memory:

```
aix> export EXTSHM=1SEG
aix> MyApp2 -m QMGR2
aix> MyApp3 -m QMGR3
aix> java -Xmx2048M MyJavaApp QMGR4
aix> unset EXTSHM
```

Or edit your login profile, for example, ~/.profile or ~/.kshrc to set EXTSHM for all applications when you log in to the system. Just add a line like this to your profile:

```
export EXTSHM=MSEG
```

Finally, in order for the EXTSHM variable to work, it must be set not only for the application trying to attach a shared memory set, but also for the application that created the shared memory set. Make sure you set EXTSHM for both applications (if they are different) or the default AIX 32-bit shared memory rules will once again take effect.

↑ Back to top

Product Alias/Synonym

WebSphere MQ WMQ

Community questions and discussion

Be the first to ask a question about this topic.

Ask a question

Document information

More support for: WebSphere MQ
Problem Determination

Software version: 5.3, 6.0, 7.0, 7.1, 7.5, 8.0, 9.0

Operating system(s): AIX

Software edition: All Editions

Reference #: 1651211

Modified date: 20 May 2014