

AIX® Support For Large Pages

Michael Mall

April 2002

Abstract

The POWER4™ processor in the IBM @server pSeries™ 690 system supports two virtual page sizes. It supports the traditional POWER architecture 4KB page size. It also supports a new 16MB page size. AIX 5L™ for POWER Version 5.1 with the 5100-02 Recommended Maintenance package contains support for 16MB size ‘large’ pages. This document describes AIX operating system support for large pages. It describes how customers may configure a POWER4 system to use large pages and how applications request large pages to back their virtual storage requirements.

Overview

Large page usage is primarily intended to provide performance improvements to high performance computing (HPC) applications. Memory access intensive applications that use large amounts of virtual memory may obtain performance improvements by using large pages. The large page performance improvements are attributable to reduced translation lookaside buffer (TLB) misses due to the TLB being able to map a larger virtual memory range. Large pages also improve memory prefetching by eliminating the need to restart prefetch operations on 4KB boundaries.

The POWER4 large page architecture requires that all virtual pages in a 256MB segment be the same size. AIX uses this architecture to support a ‘mixed mode’ process model. Some segments in a process are backed with 4KB pages and 16MB pages back other segments. Applications may request that their heap segments be backed with large pages. Applications may also request that shared memory segments be backed with large pages. Other segments in a process are backed with 4KB pages.

AIX supports large page usage by both 32- and 64-bit applications. Both the 32- and 64-bit versions of the AIX kernel support large pages.

AIX maintains separate 4KB and 16MB size physical memory pools. The customer specifies the amount of physical memory in the 16MB memory pool using the **vmtune** command. This amount of physical memory allocated to the 16MB memory pool at boot time. The remaining physical memory is used to back 4KB virtual pages. The size of the 16MB pool is fixed at boot time and cannot be changed without rebooting the system.

AIX treats large pages as pinned memory. AIX does not provide paging support for large pages. An application’s data backed by large pages remains in physical memory until the application completes. A security access control prevents unauthorized applications from using large pages. This prevents unauthorized applications from using large page physical memory and preventing authorized users from using large pages for their applications.

Large Page Application Usage

Applications may use large pages in two ways. An application may request that large pages back its data and heap segments. An application may also request shared memory segments be backed by large pages.

Large Page Data/Heap Segments

An application may request that its initialized program data, uninitialized program data (BSS), and heap segments be backed with large pages. There are two ways to request large pages back an application's data/heap segments.

1. The executable file can be marked to request large pages.
2. An environment variable can be set to request large pages.

A program's large page data/heap use is established when the program is `exec()`ed. A program cannot switch modes after it has begun executing. Large page use is inherited by children processes on `fork()`.

Marking An Executable For Large Page Use

The XCOFF header in an executable file contains a new flag to indicate that the program wants to use large pages to back its data and heap segments. This flag can be set when the application is linked by specifying the `-blpdata` option on the `ld` command. The flag can also be set or cleared using the `ldedit` command. The "`ldedit -blpdata filename`" command sets the large page data/heap flag in the specified file. The "`ldedit -bnolpdata filename`" clears the large page flag. The `ldedit` command may also be used to set an executable's `maxdata` value.

Environment Variables For Large Page Use

An environment variable is provided to allow users to indicate they want an application to use large pages for an application's data and heap segments. The environment variable takes precedence over the executable large page flag. Large page usage is provided as options on the `LDR_CNTRL` environment variable.

- `LDR_CNTRL=LARGE_PAGE_DATA=Y` specifies that the `exec()`ed program should use large pages for its data and heap segments. This is the same as marking the executable to use large pages.
- `LDR_CNTRL=LARGE_PAGE_DATA=N` specifies that the `exec()`ed program should not use large pages for its data and heap segments. This overrides the setting in a executable marked to use large pages.
- `LDR_CNTRL=LARGE_PAGE_DATA=M` specifies that the `exec()`ed program should use large pages in a mandatory mode for its data and heap segments.

Separate multiple options on the `LDR_CNTRL` environment variables by an '@' character. For example, `LDR_CNTRL=MAXDATA=0x80000000@LARGE_PAGE_DATA=Y` requests large page usage along with the `maxdata` option.

Users are advised to be cautious in their use of the environment variable to specify large page usage. Performance tests have shown there can be a significant performance loss in environments where a number of shell scripts or small, short running applications are invoked. One example saw a shell script's execution time increase over 10 times when the large page environment variable was specified. Customers are advised to only set the large page environment variable around specific applications that can benefit from large page usage.

Advisory and mandatory modes

An application can indicate it wants to use large pages for data/heap segments in either 'advisory' or 'mandatory' mode. In advisory mode, the application will use large pages if possible. The conditions needed to use large pages are:

- The userid is authorized to use large pages.
- System is running on a machine that has the POWER4 large page architecture feature.
- The customer defined a large page memory pool.
- There are enough pages in the large page memory pool to back the entire segment with large pages.

If all of these conditions are met, the application's data/heap segments will be backed with large pages. Otherwise, the application's data/heap segments will be backed with 4KB pages.

In advisory mode, an application may have some of its heap segments backed by large pages and some of them backed by 4KB pages. 4KB pages are used to back segments when there are not enough large pages available to back the segment. Executable programs marked to use large pages use large pages in advisory mode.

In mandatory mode, an application is terminated if it requests a heap segment and there are not enough large pages to satisfy the request. Customers that use the mandatory mode must monitor the size of the large page pool and ensure it does not run out of large pages. Otherwise, their mandatory large page mode applications will fail.

New 32-bit process model

32-bit applications that use large pages for their data and heap segments get a new 32-bit process model. The new model is needed due to the page protection granularity of large pages. Existing 32-bit process models mix 4KB pages with different protection attributes in the same segment. This model does not work when the protection granularity is 16MB.

The large page 32-bit process model is very similar to the existing very large process model. The model is used when the `-bmaxdata:0XXXXXXXXX/dsa` flag was specified when the application was built. The difference between the very large process model and the large page process model involves the location of privately loaded library text. In the very large process model, privately loaded library text and data are placed in the heap. In the large page process mode, privately loaded library text and data are placed in low addresses in segment 2.

The 64-bit process model with large pages is the same as the existing 64-bit process model.

Large Page Data/Heap Segments Fully Backed

The POWER4 architecture requires all pages in a segment (256MB) be backed with the same size physical pages. AIX backs the entire 256MB segment with large pages when an application requests a large page heap segment. Even if only a few bytes are needed in the new heap segment, the entire 256MB segment is backed. AIX does this to avoid terminating applications when they want to grow a heap segment (such as when using `malloc()` or `sbrk()`) and there are no large pages available to back the new space. This supports the 'advisory' mode of large page usage. It also eliminates the need for installations to closely monitor the size of their large page physical memory pools.

Using Large Pages To Back Shared Memory Segments

AIX uses the POWER4 large page architecture feature to provide large page backing for shared memory segments. Applications can request their shared memory segments be backed with large pages by specifying both the **SHM_LGPAGE** and **SHM_PIN** flags on the `shmget()` function.

The request to use large pages to back a shared segment is advisory. Large pages will back a shared memory segment under the same conditions as advisory mode large page data/heap usage. A shared segment is silently backed with 4KB pages if large pages are not available.

The physical memory to back large page shared memory and large page data/heap segments comes from the large page physical memory pool. Customers must size their large page physical memory pool to contain enough large pages for both shared memory and data/heap large page usage.

Large Page Usage Security Capability

AIX provides a security mechanism to control use of large page physical memory by non-root users. The large page physical memory pool is a fixed size, pinned memory system resource. The security mechanism prevents unauthorized users from using the large page pool and thus preventing its use by the intended users or applications.

Non-root userids must have a `CAP_BYPASS_RAC_VMM` capability in order to use large pages. A system administrator can grant this capability to a userid by the `chuser` command. The following command grants the ability to use large pages to `userid`.

```
chuser capabilities=CAP_BYPASS.RAC_VMM,CAP_PROPAGATE lpuserid
```

Both large page data/heap and large page shared memory segments are controlled by this capability.

Configuring System To Use Large Pages

The customer must configure the system to use large pages. The customer must specify the amount of physical memory to be used to back large pages. The default is to not have any memory allocated to the large page physical memory pool. The **vm tune** command is used to configure the size of the large page

physical memory pool. The following command will allocate 4GB to the large page physical memory pool. The `vmtune` command is located in the `/usr/samples/kernel` directory.

```
vmtune -g 16777216 -L 256
```

The `-g` specifies the large page size in bytes. The allowable values are 16777216 (16MB) or 268435456 (256MB). The `-L` is the number of the `-g` sized blocks that are allocated to the large page physical memory pool. While the 268435456 (256MB) size is supported by **vmtune**, on POWER4 architecture machines the storage is managed in 16MB size pages.

The customer must run the **bosboot** command and reboot before the new size large page memory pool takes affect.

Customers that want to use large pages for shared memory must enable the **SHM_PIN shmget()** system call flag. The following **vmtune** command enables the **SHM_PIN** flag. The **SHM_PIN** flag must be enabled for every system boot. Consider placing the **vmtune** command in `/etc/inittab` to have the **SHM_PIN** flag enabled automatically during system boot.

```
vmtune -S 1
```

Considerations When Determining Large Page Pool Size

Here are some things to consider when determining the size of the large page physical memory pool:

- Memory allocated to the large page physical memory pool is not available to back 4KB pages. Allocating too much physical memory to large pages will degrade system performance do to not having enough memory to back 4KB pages. During system boot, AIX reserves enough physical memory for 4KB pages to ensure that the system will boot. However, system failures may occur after booting if there is not enough physical memory to back 4KB pages.
- The size of the large page physical memory pool is fixed at boot time and remains the same for the entire boot. A reboot is required to change the size of the large page memory pool.
- Large pages are only used for applications that explicitly request them. There is no need for a large page memory pool if your applications do not request them.
- Advisory mode large page applications will use large pages if there are large pages available. If not, advisory mode large page applications will use 4KB pages. However, the inverse is not true. A 4KB application will not use large pages if the system runs low on 4KB pages.
- Mandatory mode large page applications will fail if the application requests a large page and one is not available.

Other System Changes For Large Pages

The **mprotect()** function can not be used against a large page. It returns a `-1` return code with an `EINVAL` errno if called to modify the protection attributes of a large page.

Some debug **malloc()** tools use **mprotect()** to diagnose memory management problems. These tools will not work properly with large pages. Run such applications with 4KB pages.

Pthreaded applications may use large pages for their data/heap segments. However, when large pages are used, the libpthread library does not place a protected 'red zone' page at the bottom of a pthread's stack.

The `sysconf(_SC_LARGE_PAGE_SIZE)` function call will return the large page size on systems that have large pages.

The `vmgetinfo()` function returns information about large page pools size and other large page related information.

Large Page Usage Considerations

Large page is a special purpose performance improvement feature. It is not recommended for general use. Large page usage provides performance value to a select set of applications. These are primarily long running memory access intensive applications that use large amounts of virtual memory.

Not all applications benefit by using large pages. Some applications can be severely degraded by the use of large pages. Applications that do a large number of `fork()`s (such as shell scripts) are especially prone to performance degradation when large pages are used. Tests have shown a 10-time increase in shell script execution time when the `LDR_CNTRL` environment specifies large page usage variable. Consider marking specific executables to use large pages rather than using the `LDR_CNTRL` environment variable. This limits large page usage to the specific applications that benefit from large page usage.

Consider the overall performance affect large pages may have on your system. While some specific applications may benefit from large page use, the overall performance of your system may be degraded by large page usage due to having reduced the amount of 4KB page storage available in the system. Consider using large pages when your system has sufficient physical memory such that reducing the number of 4KB pages does not significantly impact overall system performance.

Special Notices

This document was produced in the United States. IBM may not offer the products, programs, services or features discussed herein in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the products, programs, services, and features available. Any reference to an IBM product, program, service or feature is not intended to state or imply that only IBM's product, program, service or feature may be used. Any functionally equivalent product, program, service or feature that does not infringe on any of IBM's intellectual property rights may be used instead of the IBM product, program, service or feature.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed "AS IS." While IBM may have reviewed each item for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. The use of this information or the implementation of any techniques described herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. Customers attempting to adapt these techniques to their own environments do so at their own risk.

IBM is not responsible for printing errors in this publication that result in pricing or information inaccuracies.

The information contained in this document represents the current views of IBM on the issues discussed as of the date of publication. IBM cannot guarantee the accuracy of any information presented after the date of publication.

The following terms are trademarks of International Business Machines Corporation in the United States and/or other countries: IBM, AIX, AIX 5L, POWER4 and pSeries. A full list of U.S. trademarks owned by IBM can be found at: <http://www.ibm.com/ibm/licensing/trademarks/>

Other company, product and service names may be trademarks or service marks of others.