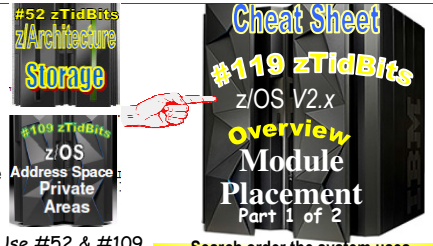
Module placement effect on application performance

- Modules begin to run most quickly when all these conditions are true:
  - They are already loaded in virtual storage
  - The virtual storage they are loaded into is accessible to the programs that call them
  - The copy that is loaded is usable
  - The virtual storage is backed by central storage (that is, the virtual storage pages containing the programs are not paged out).
- Modules that are accessible and usable (and have already been loaded into virtual storage but not backed in central storage) must be returned to central storage from page data sets on DASD or SCM.
  - Modules in the private area and those in LPA (other than in FLPA) can be in virtual storage without being backed by central storage.
    > Because I/O is very slow compared to storage access, these modules will begin to run much faster when they are in central storage.
- Modules placed anywhere in LPA are always in virtual storage, and modules placed in FLPA are also always in central storage.
  NOTE: Whether modules in LPA, but outside FLPA, are in central storage depends on how often they are used by all the users of the system, and on how much central storage is available. The more often an LPA module is used, and the more central storage is available on the system,the more likely it is that the pages containing the copy of the module will be in central storage at any given time.
- LPA pages are only stolen, and never paged out, because there are copies of all LPA pages in the LPA page data set.
  - The results of paging out and page stealing are usually the same; unless stolen pages are reclaimed before being used for something else, they will not be in central storage when the module they contain is called.
- LPA modules must be referenced very often to prevent their pages from being stolen.
- When a page in LPA (other than in FLPA) is not continually referenced by multiple address spaces, it tends to be stolen.
  - One reason these pages might be stolen is that address spaces often get swapped out (without the PLPA pages to which they refer), and a swapped-out address space cannot refer to a page in LPA.
- When all the pages containing an LPA module (or its first page) are not in central storage when the module is called, the module will begin to run only after its first page has been brought into central storage.
- Modules can also be loaded into CSA by authorized programs.
  -When modules are loaded into CSA and shared by multiple address spaces, the performance considerations are similar to those for modules placed in LPA.
  (However, unlike LPA pages, CSA pages must be paged out when the system reclaims them.)
- When a usable and accessible copy of a module cannot be found in virtual storage, either the request must be deferred or the module must be loaded.
  - When the module must be loaded, it can be loaded from a VLF data space used by LLA, or from load libraries or PDSEs residing on DASD.
- Modules not in LPA must always be loaded the first time they are used by an address space.
  How long this takes depends on:
  - Whether the directory for the library in which the module resides is cached
  - Whether the module itself is cached in storage
  - The response time of the DASD subsystem on which the module resides at the time the I/O loads the module.
- The LLA address space caches directory entries for all the modules in the data sets in the linklist concatenation (defined in PROGxx and LNKLSTxx) by default.
  - Because the directory entries are cached, the system does not need to read the data set directory to find out where the module is before fetching it.
  - This reduces I/O significantly. In addition, unless the system defaults are changed, LLA will use VLF to cache small, frequently-used load modules from the linklist. A module cached in VLF by LLA can be copied into its caller's virtual storage much more quickly than the module can be fetched from DASD.
    NOTE: You can control the amount of storage used by VLF by specifying the MAXVIRT parameter in a COFVLFxx member of PARMLIB. You can also define additional libraries to be managed by LLA and VLF.
- When a module is called and no accessible or usable copy of it exists in central storage, and it is not cached by LLA, the system must bring it in from DASD.
  - Unless the directory entry for the module is cached, this involves at least two sets of I/O operations.
    > The first reads the data set's directory to find out where the module is stored, and the second reads the member of the data set to load the module.
    > The second I/O operation might be followed by additional I/O operations to finish loading the module when the module is large or when the system, channel subsystem, or DASD subsystem is heavily loaded.
- How long it takes to complete these I/O operations depends on how busy all of the resources needed to complete them are. These resources include:
  > The DASD volume
  > The DASD controller
  > The DASD control unit
  > The channel path
  > The channel subsystem
  > The CPs enabled for I/O in the processor
  > The number of SAPs (CMOS processors only).
  NOTE: If cached controllers are used, the reference patterns of the data on DASD will determine whether a module being fetched will be in the cache. Reading data from cache is much faster than reading it from the DASD volume itself. If the fetch time for the modules in a data set is important, you should try to place it on a volume, string, control unit, and channel path that are busy a small percentage of the time, and behind a cache controller with a high ratio of cache reads to DASD reads.
- The time it takes to read a module from a load library (not a PDSE) on DASD is minimized when the modules are written to a data set by the binder,linkage editor, or an IEBCOPY COPYMOD operation when the data set has a block size equal to or greater than the size of the largest load module or, if the library contains load modules larger than 32 kilobytes, set to the maximum supported block size of 32760 bytes.



Use #52 & #109 as references for this issue.

Search order the system uses for programs

When a program is requested through a system service (like LINK, LOAD, XCTL, or ATTACH) using default options, the system searches for it in the following sequence:

1. Job pack area (JPA)
A program in JPA has already been loaded in the requesting address space. If the copy in JPA can be used, it will be used. Otherwise, the system either searches for a new copy or defers the request until the copy in JPA becomes available. (For example, the system defers a request until a previous caller is finished before reusing a serially-reusable module that is already in JPA.)

2. TASKLIB
A program can allocate one or more data sets to a TASKLIB concatenation. Data sets concatenated to TASKLIB are searched for after JPA but before any specified STEPLIB or JOBLIB. Modules loaded by unauthorized tasks that are found in TASKLIB must be brought into private area virtual storage before they can run. Modules that have previously been loaded in common area virtual storage (LPA modules or those loaded by an authorized program into CSA) must be loaded into common area virtual storage before they can run.

3. STEPLIB or JOBLIB
STEPLIB and JOBLIB are specific DD names that can be used to allocate data sets to be searched ahead of the default system search order for programs. Data sets can be allocated to both the STEPLIB and JOBLIB concatenations in JCL or by a program using dynamic allocation. However, only one or the other will be searched for modules. If both STEPLIB and JOBLIB are allocated for a particular jobstep, the system searches STEPLIB and ignores JOBLIB. Any data sets concatenated to STEPLIB or JOBLIB will be searched after any TASKLIB but before LPA. Modules found in STEPLIB or JOBLIB must be brought into private area virtual storage before they can run. Modules that have previously been loaded in common area virtual storage (LPA modules or those loaded by an authorized program into CSA) must be loaded into common area virtual storage before they can run.

4. LPA, which is searched in this order:
a. Dynamic LPA modules, as specified in PROGxx members
b. Fixed LPA (FLPA) modules, as specified in IEAFIXxx members
c. Modified LPA (MLPA) modules, as specified in IEALPAxx members
d. Pageable LPA (PLPA) modules,loaded from libraries specified in LPALSTxx or PROGxx LPA modules are loaded in common storage, shared by all address spaces in the system. Because these modules are reentrant and are not self-modifying, each can be used by any number of tasks in any number of address spaces at the same time.

Access time for modules

From a performance standpoint, modules not already loaded in an address space will usually be available to a program in the least time when found at the beginning of the following list, and will take more time to be available when found later in the list.

- The system stops searching for a module once it has been found in the search order; so, if it is present in more than one place, only the first copy found will be used. The placement of the first copy in the search order will affect how long it takes the system to make the module available.
  - Possible places are:
    1. LPA
    2. Link list concatenation (all directory entries and some modules cached automatically)
    3. TASKLIB/STEPLIB/JOBLIB (with LLA caching of the library)
    4. TASKLIB/STEPLIB/JOBLIB (without LLA caching of the library).
- For best application performance, you should place as many frequently-used modules as high on this list as you can.
- However, the following system-wide factors must be considered when you decide how many load modules to place in LPA:
  - Performance: When central storage is not constrained, frequently-used LPA routines almost always reside in central storage, and access to these modules will be very fast.
  - Virtual Storage: How much virtual storage is available for address spaces that use the modules placed in LPA, and how much is available for address spaces that do not use the modules placed in LPA.

Module placement effect on system performance

- Whether the placement of a module affects system performance depends on how many address spaces use the module and on how often the module is used.
  - Placement of infrequently-used modules that are used by few address spaces have little effect on system-wide performance or on the performance of address spaces that do not use the modules.
  - Placement of frequently-used modules used by a large number of address spaces, particularly those used by a number of address spaces at the same time, can substantially affect system performance.

Placement of modules in LPA:

- More central storage can be used when a large number of address spaces each load their own copy of a frequently-used module, because multiple copies are more likely to exist in central storage at any time.
  NOTE: One possible consequence of increased central storage use is increased paging.
- When frequently-used modules are placed in LPA, all address spaces share the same copy, and central storage usage tends to be reduced.
  - The probability of reducing central storage usage increases with the number of address spaces using a module placed in LPA, with the number of those address spaces usually swapped in at any given time, and with how often the address spaces reference the module.
  NOTE: You should consider placing in LPA modules used very often by a large number of address spaces.
- By contrast, if few address spaces load a module, it is less likely that multiple copies of it will exist in central storage at any one time.
  - The same is true if many address spaces load a module, run it once, and then never run it again, as might happen for those used only when initializing a function or an application.
  - This is also true when many address spaces load a module but use it infrequently, even when a large number of these address spaces are often swapped in at one time; for example, some modules are used only when unusual circumstances arise within an application.
  NOTE: Modules that fit these descriptions are seldom good candidates for placement in LPA.
- You can add modules to LPA in these ways:
  - Add the library containing the modules to the LPA list
    > Any library containing only reentrant modules can be added to the LPA list, which places all its modules in LPA.
    NOTE: Modules are added to LPA from the first place in the LPA list concatenation they are found.
  - Add the modules to dynamic LPA
    > Use this approach instead of placing modules in FLPA or MLPA whenever possible.
    > Searches for modules in dynamic LPA are approximately as fast as those for modules in LPA, and placement of modules in dynamic LPA imposes no overhead on searches for other modules.
    NOTE: Another LPA module whose address was stored before a module with the same name was loaded into dynamic LPA might continue to be used.
  - Add the modules to FLPA
    > Modules in the fixed LPA list will be found very quickly, but at the expense of modules that must be found in the LPA directory.
    > It is undesirable to make this list very long because searches for other modules will be prolonged.
    NOTE: Modules placed in IEAFIXxx that reside in an LPA List data set will be placed in LPA twice, once in PLPA and once in FLPA.
  - Add the modules to MLPA
    > Like placement in FLPA, placing a large number of modules in MLPA causes searches for all modules to be delayed, and this should be avoided.
    > The delay will be proportional to the number of modules placed in MLPA, and it can become significant if you place a large number of modules in MLPA

Placement of modules outside LPA:

- In addition to the effects on central storage, channel subsystem and DASD subsystem load are increased when a module is fetched frequently from DASD.
  - How much it increases depends on how many I/O operations are required to fetch the module and on the size of the module to be fetched.

General virtual storage allocation considerations

- Virtual storage allocated in each address space is divided between the system's requirements and the user's requirements. The base system control programs require space from each of the basic areas.
- Storage for SQA, CSA, LSQA, and SWA is assigned for either the system or a specific user. Generally, space is assigned to the system in SQA and CSA and, for users, in LSQA or SWA.

5. Libraries in the linklist, as specified in PROGxx and LNKLSTxx.
By default, the linklist begins with SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, and SYS1.SIEAMIGE. However, you can change this order using SYSLIB in PROGxx and add other libraries to the linklist concatenation.