Module placement effect on virtual storage

- When a module is loaded into the private area for an address space, the region available for other things is reduced by the amount of storage used for the module.
- Modules loaded from anywhere other than LPA (FLPA, MLP) can be loaded into any address space or into CSA.
- If a module is loaded below LPA below 16 megabytes, the expansion of the explicitly-allocated common area below 16 megabytes will be increased by the amount of storage used for the module.
- When the explicitly-allocated common area does not end on a segment boundary, MLP cannot determine the exact size of the expanded area.

On the other hand, virtual storage fragments and changes when modules are added to LPA. Depending on whether a segment boundary is crossed or not.

When modules are added to an LPA below 16 megabytes, and this does not result in the expansion of explicitly allocated common storage past a segment boundary, less virtual storage is required for virtual storage (as page overflow) storage.

- The amount of CSA and SQA specified in EASYmx will still be available, but the amount of SQA specified for the current storage class might change.

When the addition of modules to LPA does not result in a reduction in the size of the private area below 16 megabytes, adding load modules to LPA increases the amount of private area available for address space management. This is because the system uses the copy of the load module in LPA rather than from segmental storage. Therefore, with the amount of usable storage removed from the private area.

When modules are added to LPA below 16 megabytes, the growth in LPA can cause the common area below 16 megabytes to cross one or more segment boundaries, which will reduce the available private area below 16 megabytes by a corresponding amount; each time the common area crosses a segment boundary, the private area is reduced.

- For example, if an address space uses 1.5 megabytes of modules, all of them are placed in LPA up to 16 megabytes, and this causes the common area to expand across two segment boundaries. 5 megabytes less private storage area will be available for programs in that address space.

NOTE: If adding the same 1.5 megabytes causes only one segment boundary to be crossed. Only 2.5 megabytes more storage area is required. Therefore, it would cause no change in the amount of private area available to programs in those address spaces. This assumes no other changes made to any other common storage.

When the size of the private area is reduced as a result of placing modules in LPA below 16 megabytes, less storage area is required than from segmental storage. A process similar to that described for LPA is also true when using VIO data sets. The amount of storage used for VIO data sets, all other data types can be taken out of VIO data sets.

- For example, you might use 16 megabytes of storage by specifying module packing through the IEAPAKxx member of the PARM.

- You can specify VIO data sets in the IEAPAKxx member. Upon LPA expansion, you can use the IEAPLKxx member and the IEAMEMxx member.

- If the first PLPA page data set specified during IPL is large enough, all PLPA pages are written to the same data set.

- If the second PLPA page data set specified during IPL is large enough, the remaining pages are written to the common page data set (the second one specified during IPL). You can choose the first and second PLPA page data sets as long as they are legible, all PLPA pages are written to the common page data set.

- Any module in the modified private area of the address space can be placed in the LPA list, place only the most frequently-used TSO/E modules on your system in dynamic LPA.

- The system will try to load with the last 20 free areas available.

- To reduce the possibility of requiring reserved virtual storage, space, integrity, or integrity, the presence of a high percentage of frequently-used reentrant modules (and containing modules that are not in the list) in the IPA.

- For example, if TSO/E response time is too high and average consideration stores it, add the CMU to the IPA.

- Make sure you do not inadvertently duplicate modules, modules, name, or aliases that already exist in LPA.

- Use local load modules (LIPA) and the CMU to the IPA.

- Use dynamic LPA to do this rather than MLP whenever possible.

- Module loading might have to be modified when a SETPROG command is executed.

To load modules in dynamic LPA, list them on an LPA ADD statement in a PROG command processors on a system where TSO/E response time is not important, and low-use work processors when this performance problem is critical.

Virtual storage is a finite resource, and placement of modules in LPA should be prioritized when necessary.

- Leave low-use modules from the list (such as those in CMU) on systems where TSO/E performance is not critical) and low-use applications modules outside user submission limits to allow for other modules to serve user submission limits.

NOTE: This placement usually has little or no effect on other address spaces that do not use these modules.

- Configure get-of-aid storage on central storage.

- If other measures (like WLM policy changes, managing the content of LPA, and balancing central and expanded storage allocations) fail to control storage saturation, and paging and swapping begin to affect your critical workloads, the most effective solution may be to add virtual storage to the system.

- Sometimes, this is as simple as changing the storage allocated to different LPARs on the same processor.

- You should consider other options only when you cannot add storage to the system.

- Use the appropriate storage-class memory (SCM) on FlashExpress solid-state drives (SSD) as a second type of auxiliary storage. Auxiliary storage is required.

- If you experience significant PLPA paging, you can use the fixed LPA to reduce page-fault overhead and increase performance at the expense of possible page faults.

- You can assure that specific modules are kept in central storage by adding them to the fixed LPA by listing them in IEFIXxx.

- This trade-off is desirable with a system that is CPU-bound, where it can be best to divert some of the central storage from possible use by additional address spaces, and use it for additional LPA modules.

- If you have insufficient PLPA paging, you can use the fixed LPA to reduce page-fault overhead and increase performance at the expense of possible page faults.

- You can assure that specific modules are kept in central storage by adding them to the fixed LPA by listing them in IEFIXxx.

- This trade-off is desirable with a system that is CPU-bound, where it can be best to divert some of the central storage from possible use by additional address spaces, and use it for additional LPA modules.

- Your HPix modules probably need to be listed in IEFIXxx because they tend to be referenced frequently enough to remain in central storage.

- A large LPA makes less central storage available for pageable programs.

- Accordingly, fewer address spaces might be in central storage than would otherwise be the case.

- In loss in throughput should occur, however, if CPU use remains reasonably high.

- NOTE: A large LPA can increase other demand paging and swapping activity and this will impede the system's normal self-tuning actions because keeping these modules in storage might prevent other, more frequently-used modules, from being in storage. This works when workloads shift over the course of time. Also, like module packing lists, fixed LPA lists need to be maintained when installing new releases of software, installing significant amounts of service, or when your workloads change. If you can prevent LPA paging by adding central storage, the system will be simpler to manage.

- When using the IEAPAKxx member for PLPA paging, you can use the IEAPAKxx member for PLPA paging.

- You can minimize page faults and read arm movement by specifying module packing through the IEAPAKxx member of parmlib.

- Module packing reduces page faults by making the thickest page fault those small modules (less than 4KB bytes) that refer to each other.

- Module groups that refer to each other but that exceed 4KB bytes in combined size can be placed in adjacent (4K) auxiliary storage slots to reduce seek time.

- Thus, use IEFAPAXxx should improve performance compared to the simple loading of the PLPA from the LPALST.

- You must maintain module packing lists whenever you install new versions of software or significant service, or when your workloads change.

- You can increase the amount of central storage enough to control PLPA paging rather than using a module packing list, the equivalent of using this service.

- If the first PLPA page data set specified during IPL is large enough, all PLPA pages are written to the same data set.

- If the first page data set is not large enough to contain all PLPA pages (for example, when allocated as a one-cylinder data set as recommended below), the remaining pages are written to the common page data set (the second one specified during IPL).

For best performance, all PLPA pages would be written to a single page data set on a single DASD volume. Performing significant subtasks in one data set would reduce the number of I/O operations and free the cylinder and define enough storage for the common page data set to contain both the PLPA and common pages. When defined this way, the PLPA and common page data sets should be contiguous, with the small PLPA data set data set followed immediately by the large common page data set on the volume.

You should consider allocating these data sets this way unless you experience significant I/O delays. If you use the PLPA, it is not possible to change the size of the data set, and it may also limit the number of I/O operations for which space must be managed and simplified.

- If you have a very large number of page faults or paging activity that affects critical applications, and you cannot control central or storage-class memory (SCM) to manage, you can tune the page subspace.

- When most paging subspace activity is for swapping, a large number of page data sets can outperform a small number of page data sets, even on high-speed or cached devices.

- If you have substantial swapping, consider using eight or more page data sets on different low-use volumes on low-use control units and channel paths. However, these should generally be considered stop-gap solutions.

- If the storage demand continues to grow, tuning the paging subspace will usually delay the inevitable for only a short time.

- In the long run, adding central storage is always a better solution.

NOTE: Some cached devices also do not support cached paging.