Pinning data in storage from past z/OS Customer Class

There are many ways to pin data (objects) in storage with the mainframe. One means is to use virtual lookaside facility (VLF) which is a set of services that can improve the performance and response time of your applications that must retrieve a set of data for many users. VLF creates and manages a data space to store an application's most frequently used data. When the application makes a request for data, VLF checks its data space to see if the data is there. If the data is present, VLF can rapidly retrieve it without requesting I/O to DASD. To take advantage of VLF, an application must identify the data it needs to perform its task. The data is known as a data object. Data objects should be small to moderate in size, named according to the VLF naming convention and associated with an installation-defined class of data objects. VLF is intended for use with major applications. Because VLF runs as a started task that the operator can stop or cancel, it cannot take the place of any existing means of accessing data on DASD. Any application that uses VLF must also be able to run without it.
Note: There are PARMLIB(COFVLFxx) changes required to use VLF.

Decide when to use VLF -

Before creating a new application or recoding an existing one to take advantage of VLF, you must answer a few questions about the data objects your application will use.

- **What kind of data objects does your application work with?**

VLF works best with two kinds of data: (1) data objects that are members of partitioned data sets, located through a partitioned data set (PDS) concatenation, or (2) data objects that, while not PDS members, could be easily described as a collection of named objects that are repeatedly retrieved by many users. If neither description fits your data objects, it is likely that you would not obtain any performance benefit from VLF. An application that retrieves named shared objects might save development cost by using VLF, but there are storage overhead costs associated with using VLF, and it is best used when you expect performance benefit.

- **Is your data accessed frequently enough so that much of it is likely to remain in processor storage?**

Like data in your address space private storage, data stored through VLF is subject to page stealing. Thus, appropriate data for VLF is data shared by many users, so that the combined reference rate of all users is likely to keep the data in processor storage. That is, VLF works best when a significant portion of the data is likely to remain in processor storage and not be paged out to auxiliary storage.
How large are your data objects?
VLF works best with relatively small objects because less virtual storage is expended to reduce the number of I/O operations. Very large objects, if they are not used frequently enough to remain in central or expanded storage(not sure whether you still use expanded stg), might take longer to retrieve through VLF than through traditional I/O from DASD.
How long do you expect users to actively retrieve objects through your application?
VLF processing adds overhead to the initial retrieval of an object from DASD, and your application must identify users to VLF. Some applications might require more identifications than others. If many retrieves are not likely for each identification,

then the relative benefit of using VLF is less.

Other considerations important to answer when deciding whether or not to use VLF are:

- **Is your system storage-constrained?**

VLF is designed to improve performance by increasing the use of virtual storage to reduce the number of I/O operations. For a system that is already experiencing a central, expanded, or auxiliary storage constraint, this strategy is probably not a good choice.
Is user access to data controlled at the address space level or at the task level? When your application identifies an end user to VLF, VLF returns a user token (identifier). Any task in the user address space can then use that token. Thus, VLF checks access to data objects at the address space level. You cannot use VLF to manage data that must be restricted to only certain tasks in an address space.
Is running in either supervisor state or with PSW key mask 0-7 a problem for your application? This environmental restriction is essential to maintain system integrity. In addition, if your application obtains its VLF data objects from DASD, you must be sure that the program that reads in the object and creates the VLF object maintains the integrity of the data.
Does your application process the data obtained from DASD in a repetitive way before passing it to each user? If it does, you can potentially obtain additional benefit from VLF by saving preprocessed objects in VLF storage rather than exact copies of the DASD objects. This technique would require some recoding for an existing application, but it could provide a significant benefit if the amount of common repetitive processing is large.