CA PMO™ Runtime Performance Optimizer

Product Guide

r4.4

CA Technologies Product References

This document references the following CA Technologies products:

- CA ASTEX® Performance (CA ASTEX)
- CA PMO™ Runtime Performance Optimizer (CA PMO)
- CA NetSpy® Network Performance (CA NetSpy)
- CA QuickFetch® Runtime Performance Optimizer (CA QuickFetch)
- CA Roscoe® Interactive Environment (CA Roscoe)
- CA Auditor for z/OS (CA Auditor) (formerly known as eTrust® CA-Examine® Auditing)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Understanding CA PMO

This chapter introduces CA PMO Runtime Performance Optimizer (CA PMO) and describes its many features.

System programmers installing and using this product should have a solid understanding of z/OS operating system internals, subsystems, and components. System operators should understand the hardware controls and general organization of the z/OS operating system.

Note: References to the z/OS operating system throughout the CA PMO documentation pertain to the supported releases of z/OS operating systems.

This section contains the following topics:

Overview

CA PMO is designed to improve the performance of z/OS operating system.

CA PMO enhances the way that z/OS ordinarily searches for *members* of a *partitioned data set* (PDS). A PDS is a collection of files, called members. A member can contain any kind of data, including executable code, CLISTs, JCL, macros, ISPF panels, or ASCII text. In addition to members, a PDS has a *directory* that contains the location of each member in the PDS. Another name for a PDS is a *library*.

Note: With DFP 3.2 and above, IBM enabled a new format of a PDS called PDSE. CA PMO provides limited toleration support for concatenations containing PDSE data sets. For more information, see CA PMO and PDSE-type Libraries in this chapter.

When a member of a library must be found, z/OS looks in the directory of that library for the location of the member. This search procedure is efficient if the directory of the library that contains the member is in processor storage (the CPU). However, in most cases the library directory is not in processor storage, and z/OS must conduct the search on a Direct Access Storage Device (DASD), which is an external storage device. A substantial number of system-related programs, such as compilers, sort programs, and TSO command processors, are not resident in processor storage. These programs must be searched for on DASD every time they are needed. In addition, all libraries that users own must be searched for on DASD.

Directory searches that use I/O to DASD are slower than those conducted in processor storage. Too much DASD I/O can erode system performance and can become particularly troublesome with the following types of directory searches:

- When the library is large and its directory occupies several DASD tracks.
- When the search involves examining the directories of several concatenated libraries. This situation is typical, because even the simplest activity for a user (such as invoking a text editor) forces z/OS to read long lists of program libraries.
- When two or more systems share one DASD this causes contention delays on several systems at once.
- When a search for a frequently requested member is directed against a JOB, STEP, or TASK library, even though the member actually resides in the Link Pack Area (LPA) or LNKLST. This can cause excessive I/O for the library. The impact of this type of search is addressed more fully in the section on Hash Tables.

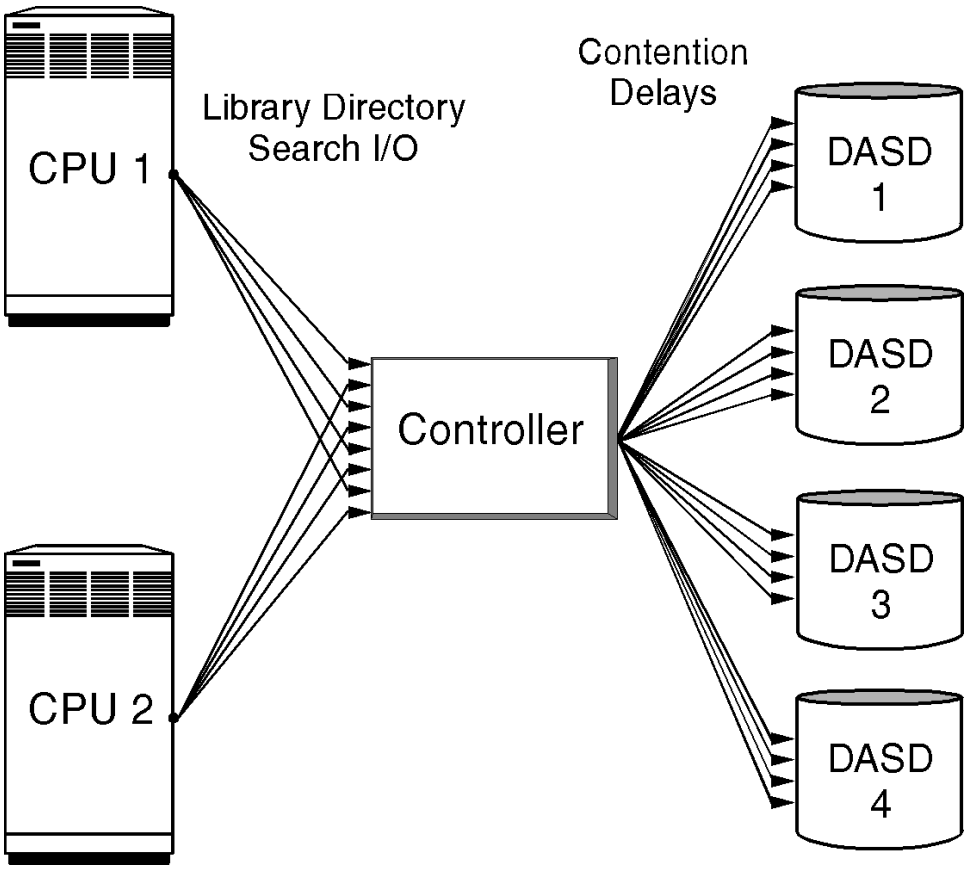
In addition, the channel program used for a search cannot use rotational position sensing (RPS). The channel, control unit, head-of-string, and DASD are busy during an entire search. When an entire path is busy, it can cause delays for other tasks sharing that path.

More Information:

CA PMO and PDSE-type Libraries (see page 24)

Directory Search I/O Without CA PMO

Directory searches for a library member typically involve a substantial amount of DASD I/O, as shown in the following illustration:

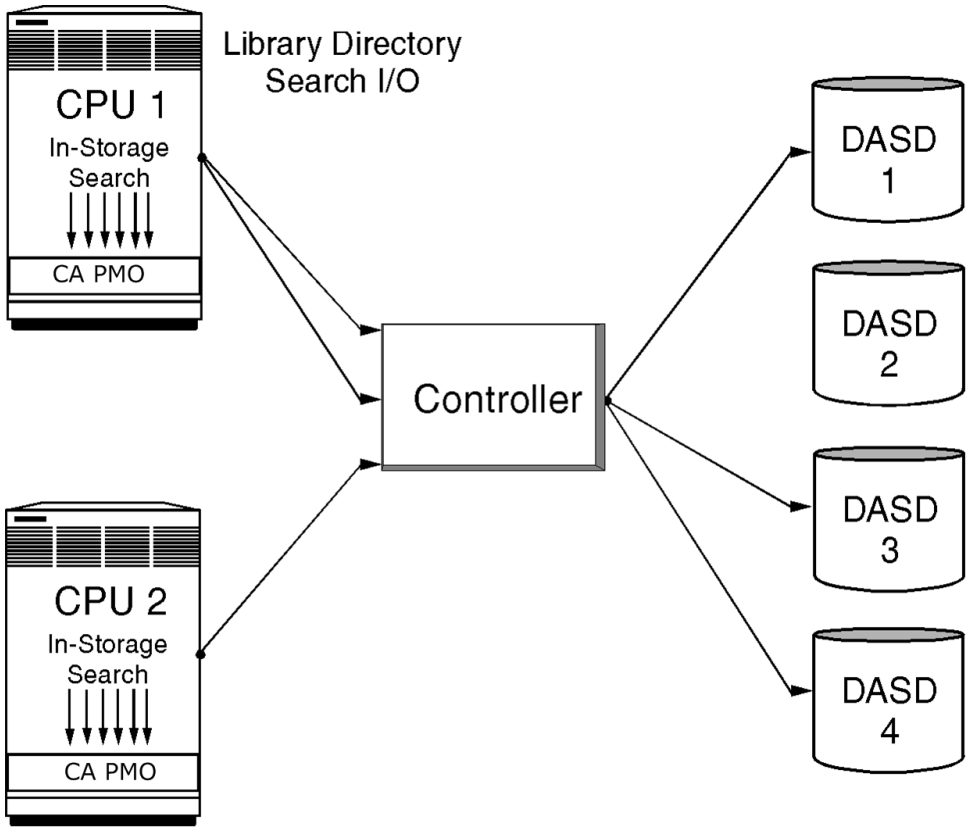


CA PMO assists with directory searches by automatically determining what libraries are having DASD I/O directory searches, and creating and maintaining lists of these libraries' directories in processor storage. CA PMO in-storage lists allow it to resolve directory searches of these libraries without resorting to DASD I/O. You can specify to CA PMO the libraries that it either must or must not handle; however, CA PMO can determine what libraries to manage without any assistance from a user.

Directory Search I/O With CA PMO

CA PMO automatically reduces or eliminates the DASD I/O associated with library directory searches, as shown in the following illustration. When CA PMO is active, most or all DASD directory search I/O is replaced by in-storage search.

Equation 1: Show the directory search !/O with CA PMO



What You Need to Know About CA PMO

To understand CA PMO, you should be familiar with the following terms:

BLDL, FIND, and DESERV GET requests

On z/OS systems, library directory searches are initiated by a Build List (BLDL) macro instruction, a FIND (Type D) macro instruction, or a DESERV GET function call. Collectively, these search initiators are called *directory search requests*. Directory search requests are issued by programs, applications, and by a component of z/OS called Program Management. When z/OS receives a directory search request, CA PMO intercepts the request. Rather than allowing z/OS to conduct its usual procedures for library directory searches, CA PMO attempts to resolve the search efficiently.

LNKLST library

Directory search requests may be for members of LNKLST libraries. Each LNKLST library may be either a PDS or PDSE, and is available to everyone on the system to use. The entire LNKLST consists of a concatenation of these libraries. These libraries include operating system programs, compilers, TSO command processors, and other programs that many users would access. When the operating system is initialized, the LNKLST is defined as the default location for these programs. (Specifically, the LNKLST is the default location for nonresident program load modules.)

Private library

Directory search requests may be for members of private libraries as well as for LNKLST library members. A private library is any PDS or PDSE that is not in the LNLKST. In other words, private libraries are partitioned data sets that users create, edit, move, copy, and delete on a daily basis.

Library management

When a directory search request is issued for a member of a private or LNKLST library, CA PMO handles the resulting library directory search for that member. This basic function of CA PMO is referred to as *library management*. CA PMO performs library management using two basic components: a *managed list*, which contains directory entries for LNKLST library members, and a set of *hash tables*, each of which contains an entire private library directory.

CA Common Services for z/OS

CA Common Services for z/OS are a common set of services that may be used by any z/OS CA product. These services are maintained separately from the product and are documented and installed separately as well. CA PMO uses CAIRIM for installation services and security.

Licensing Management Program (LMP)

CA PMO also uses CAIRIM services to determine product licensing authorization.

CA Health Checker

Provides a simple and consistent method for CA products to create health checks to run under the IBM Health Checker for z/OS. The IBM Health Checker for z/OS helps you identify potential problems in your z/OS environment by checking system or product parameters and system status against recommended settings. CA has joined other vendors in creating checks for CA z/OS products. CA PMO health checks are automatically activated on the target system when the product is started on a system where the following components are installed and configured:

- CA Health Checker Common Service
- IBM Health Checker for z/OS

For more information on installing the CA Health Checker Common Service, see the *CA Common Service Installation Guide.*

For more information about the IBM Health Checker for z/OS, see the *IBM Health Checker for z/OS User Guide*.

The Managed List

The CA PMO managed list is a set of LNKLST directory entries. LNKLST libraries consist of members that everyone on the system can use, such as executable code for popular databases, word processing programs, compilers, sort programs, and so on. The managed list resides in the Extended Common Service Area. The salient features of the managed list are as follows:

- The entries in the managed list reflect the most frequently requested LNKLST members on the system.
- The size of the list is kept small, which reduces the time that CA PMO needs to find a particular entry.
- CA PMO sorts the managed list frequently to ensure that the most-used members on the system stay at the top of the list. This decreases search times because CA PMO scans the managed list from the top down.
- CA PMO also keeps entries in the managed list for LNKLST members that z/OS looked for, but could not find. For instance, if a user types a CLIST member name from TSO, but forgets to type a percent sign (%) before the name, z/OS looks in the LNKLST for the member and does not find it.CA PMO places an entry for the CLIST member name in the managed list. Subsequently, CA PMO can determine that the member will not be found in the LNKLST, without involving z/OS in the search.
- CA PMO revises the managed list continually to keep its entries consistent with directory search activity at the current time. For instance, CA PMO may manage a batch program for producing financial reports from evening until morning, when that program is accessed frequently. In contrast, CA PMO may manage a graphics editor during regular working hours.

By default, CA PMO manages members from all PDS LNKLST libraries in the managed list. Members from LNKLST libraries that are PDSE are *not* managed in managed lists (see the section CA PMO and PDSE-type Libraries in this chapter). However, you can instruct CA PMO to exclude certain libraries or sets of libraries from management. You may specify the exclusions with parameters that CA PMO processes when it starts up, and you can exclude libraries through commands issued at the console while CA PMO is running. You can also designate a maximum number of members that CA PMO may manage in the managed list. If that number is reached, CA PMO may stop managing an infrequently accessed member and replace it with a more frequently used one.

By using the managed list instead of the default z/OS facilities, CA PMO reduces DASD I/O and increases the speed of searches for LNKLST members. In general, CA PMO resolves over 90% of all BLDL/FIND requests for LNKLST library members.

Can Replace Library Lookaside (LLA) Facility

The managed list supplements or replaces the Library Lookaside (LLA) facility that resides on z/OS.

To prevent DASD I/O for frequently requested load modules, LLA keeps them in a Virtual Lookaside Facility (VLF) data space. Requests for these members are resolved from the VLF data space.

If you use CA PMO and CA QuickFetch you can discontinue use of LLA and remove load modules from VLF. LLA requires more central storage than the managed list. CA QuickFetch reduces DASD I/O for fetching executable code, or load modules, from DASD. CA PMO will automatically handle updates without the manual intervention and delays associated with LLA.

If you continue to use LLA, CA PMO can automatically perform selective refreshes of LLA, updating only the members or libraries that were updated on the system, or CA PMO can warn the operator that a refresh is needed. When you use CA PMO, LLA pages are referenced less frequently, since CA PMO resolves most of the searches that LLA would have resolved.

Note: As discussed on the following pages, CA PMO manages private library directories in hash tables. If you use both LLA and CA PMO, we recommend that you allow CA PMO to manage all private libraries, and that you do not allow LLA to manage any of them.

Hash Tables

CA PMO hash tables are lists of directories for libraries that are not in the LNKLST. Typically, these libraries belong to users. In CA PMO terminology, these libraries are called *private libraries*.

Note: LNKLST libraries that are accessed outside the LNKLST concatenation, perhaps in a JOB/STEP/TASK library concatenation, can also be managed in hash tables.

Private libraries can be any type of partitioned data set. They may contain code, documentation, spreadsheets, CLISTS, JCL, ISPF panels, macros, and products of other popular applications. The hash tables reside in the extended private region.

Because each hash table contains the entire directory of a private library, CA PMO can use a hash table to resolve a search for a member whether or not the member is found. This eliminates I/O library directory searches both for members that can be found and for those that cannot be found.

As with the managed list, CA PMO automatically selects the libraries that it manages in the hash tables. CA PMO bases its initial management decisions on directory search activity when it starts up. It continually revises its list of libraries to be managed, adding new private libraries for management as warranted by system demand. You can specify the maximum number of private libraries that CA PMO may manage. You can also limit the amount of virtual storage in the CA PMO address space, which limits the number of libraries that CA PMO can manage. CA PMO may stop managing certain private libraries in favor of others when the maximum number of libraries has been reached, or if its virtual storage space is exhausted.

CA PMO commands and parameters allow you to fine-tune its management of private libraries. If you wish, you can give CA PMO a list of private libraries that it may consider eligible for management, or you can tell CA PMO to never manage a particular set of private libraries. You can specify a generic description of the private libraries to be managed or to be excluded from management. For instance, you can instruct CA PMO to exclude all libraries that have names beginning with USER, or you can include all libraries on a particular DASD volume.

If you have a severe paging problem on your system, you can manually specify the private libraries that CA PMO will manage instead of allowing CA PMO to select them automatically. Methods for handling paging constraints are addressed in the installation and tuning instructions in this manual.

As with the managed list, the hash tables allow CA PMO to reduce DASD I/O and to increase the speed of searches for members. When CA PMO manages private libraries automatically, it can resolve over 85% of all directory searches for members of private libraries.

Examples of How the Hash Tables Eliminate DASD I/O

Example 1

A BLDL/FIND request is issued for a module named PDS5PGM. It resides in library PDS5, which is the last library in a concatenation of libraries:

```
//          EXEC  PGM=PDS5PGM
//STEPLIB   DD    DSN=PDS1
//          DD    DSN=PDS2
//          DD    DSN=PDS3
//          DD    DSN=PDS4
//          DD    DSN=PDS5
```

Library directories are searched in the order in which the libraries appear in the concatenation. As a result, PDS1, PDS2, PDS3, and PDS4 always are searched before PDS5. If CA PMO manages all of these libraries in hash tables, DASD I/O is avoided.

Example 2

A BLDL/FIND request is issued for a module named LPAPGM. It resides in LPA:

```
//          EXEC  PGM=LPAPGM
//STEPLIB   DD    DSN=PDS1
//          DD    DSN=PDS2
//          DD    DSN=PDS3
```

Under some circumstances, LPAPGM can be retrieved from LPA without any DASD directory searches. But in z/OS systems, JOB/STEP/TASK libraries are searched before LPA. Ordinarily, if PDS1, PDS2, and PDS3 were JOB/STEP/TASK libraries, they would require DASD directory searches. But if CA PMO manages these libraries in hash tables, DASD I/O is avoided.

For information about standard search orders under z/OS, see the chapter "How CA PMO Optimizes Directory Searches" in the CA PMO *Systems Programmer Guide*.

Example 3

A BLDL/FIND request is issued for a module named LKLSTPGM. It resides in a LNKLST library. Either LKLSTPGM has an entry in the managed list or LLA is active:

```
//         EXEC PGM=LKLSTPGM
//STEPLIB  DD  DSN=PDS1
//         DD  DSN=PDS2
//         DD  DSN=PDS3
```

Under some circumstances, LKLSTPGM can be retrieved without DASD I/O. But on z/OS systems, JOB/STEP/TASK libraries are searched before LNKLST libraries. If PDS1, PDS2, and PDS3 were JOB/STEP/TASK libraries, they would have I/O directory searches. But if CA PMO manages these libraries in hash tables, DASD I/O is avoided.

Updates to Libraries and Members

CA PMO quickly detects updates to members and libraries. When it detects an update, it selectively refreshes its storage areas; only the entries for the updated members or libraries are refreshed. If you have a multi-processor site, CA PMO handles updates through the optional components CA PMO Cross-System (PMO/XSYS) and CA PMO XCF Enhanced Data Transmission Component (PMO/XCF). PMO/XSYS and PMO/XCF are distributed as a standard part of CA PMO. When a member is updated, PMO/XSYS or PMO/XCF informs each instance of CA PMO on other systems in your network of the update. PMO/XCF uses XCF messaging to distribute the information within a sysplex. PMO/XSYS has a control file on shared DASD, and can be used for communication across sysplexes.

Member updates include link-edits; renaming, adding, or deleting members; and member modifications such as editing an existing member. After an update to a member occurs, CA PMO modifies the member hash table entry or deletes its entry from the managed list, as appropriate. CA PMO replaces the managed list entry upon the next BLDL/FIND request for the member.

Library updates include:

- Compressing a library
- Copying one or more members into a library with IEBCOPY
- Moving a library
- Restoring, compressing, or moving a library with utilities such as FDRABR or DFDSS

Moving a library may involve:

- Deleting a library and then reallocating it
- Deleting a library and then renaming another library with the name of the deleted library

Following an update to a LNKLST library, CA PMO deletes the entries for that library from the managed list, and re-includes them upon later BLDL/FIND requests for them.

Because CA PMO processes LNKLST library updates automatically, if you refresh LLA often, you should consider discontinuing use of LLA when you are running CA PMO. If you use LLA with CA PMO, CA PMO can automatically issue an LLA refresh, or it can warn the operator when a refresh is needed. If a LNKLST library is updated on another system on your network, PMO/XSYS or PMO/XCF performs the refresh. After the LLA refresh, CA PMO monitors BLDL/FIND requests for LNKLST members. When the requests are issued, CA PMO replaces the members' directory entries in the managed list.

Note: As long as you follow standard z/OS conventions for updating a library, CA PMO handles all updates correctly. When you compress or move a library, be sure there is no directory search activity against it. This is particularly important when the library resides on another system from the one on which the update occurs. (PMO/XSYS can take a few seconds to communicate these updates to the other systems that are running CA PMO, however, when using PMO/XCF in a sysplex, the update will be communicated instantly). When you move a library, be sure to stop all other applications that are using that library. For more information on library and member updates, see the chapter "Updating Libraries" in the *CA PMO Systems Programmer Guide*.

CA PMO and System Managed Storage

System Managed Storage (SMS) is an architecture from IBM that helps manage increasingly complex storage resources by reducing the amount of active management that data center personnel have to perform. It is designed to accommodate the growth of your storage subsystem, letting you manage storage resources more easily.

SMS helps manage the performance of your storage resources by:

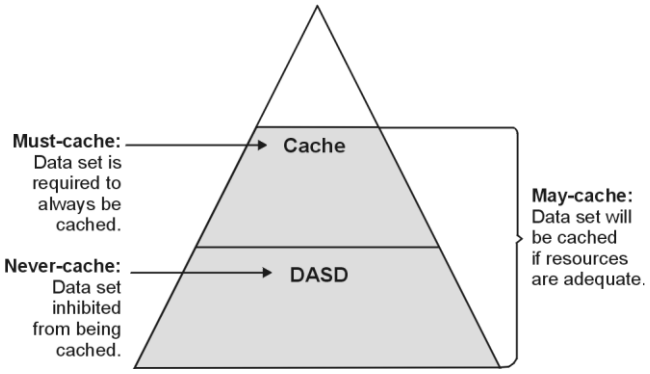
- Using multiple levels of the storage hierarchy (primarily cache controller memory) to provide higher levels of performance
- Placing data sets at an appropriate level of the storage hierarchy at allocation time, depending on performance requirements

How SMS Places Data in the Storage Hierarchy

In an SMS environment, storage administrators must develop SMS constructs, including data classes, storage classes, and management classes that identify data set characteristics and policies as well as ACS routines. When a data set is allocated, SMS places it in a level of the storage hierarchy determined by the *storage class* and *storage group* assigned to it by the ACS routines.

Under SMS, a data set is assigned to one of three *storage class types* or caching statuses based on the millisecond response (MSR) objective that the system administrator assigns to the data set. The three classes are:

- Must-cache
- May-cache
- Never-cache



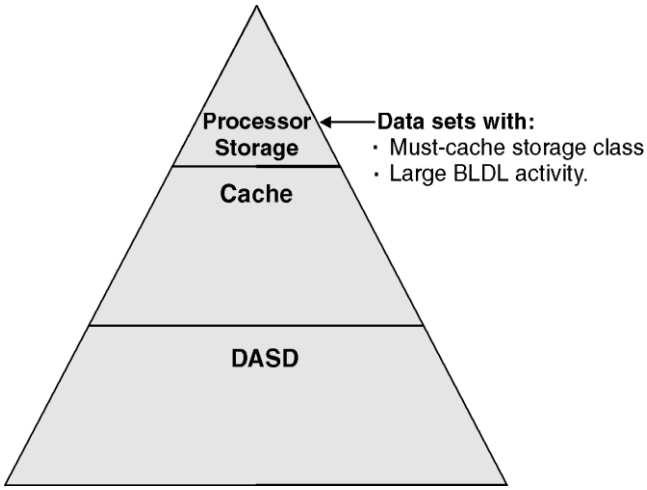
How CA PMO Enhances SMS Functionality

CA PMO automatically identifies SMS-managed data sets and their storage classes, and gives priority to data sets in the must-cache storage class. The directories of must-cache data sets in the CA PMO candidate list are placed in processor storage immediately. If resources are constrained, must-cache data sets remain in processor storage while other data sets are removed.

Under SMS, when a must-cache data set is concatenated behind a never-cache data set, SMS will not cache it. CA PMO, on the other hand, enhances performance by automatically attempting to bring the data set directory into processor storage. For more information, see the section How Storage Classes are Represented at I/O Time by SMS in the chapter "How CA PMO Works" in the *Systems Programmer Guide.*

In summary, the benefits CA PMO offers in an SMS environment are:

- Expanding usage of the storage hierarchy by taking advantage of processor storage, as illustrated in the following graphic
- Freeing up valuable cache controller resources
- Lowering device utilization
- Helping ensure that SMS response time objectives are met



CA PMO Benefits

CA PMO increases job throughput on your system, it reduces the time needed to process batch jobs, and it improves response time for your users. CA PMO prolongs the time that the system is useful, allowing you to defer hardware upgrades. Since CA PMO saves time for your users and for the CPU, it also saves money.

Note: You can use CA NetSpy Performance to display user response time and its two major components: host time and network time.

Other benefits include:

- Reduced EXCPs and I/O to DASDs with libraries. This improves response time, and it lets you allocate more I/O resources to paging, database accesses, and other functions that can impair the performance of your system.

  Note: CA ASTEX Performance can help you determine where to move data sets to make the best use of the capacity of your system.
- Reduced I/O for directory searches to your hardware cache, which lets you devote the hardware cache to other types of I/O.
- Reduced need for clone libraries, which saves DASD resources and administrative time.
- Decreased busy time for certain channels, control units, heads-of-string, and devices.
- Fewer manual operations for handling updates. CA PMO automatically detects same-system and cross-system updates, and it selectively refreshes its in-storage areas in response to updates.
- Discontinued use of Library Lookaside (LLA). CA PMO saves processor storage and provides complete, reliable, and automatic updates. Alternatively, CA PMO can automatically refresh LLA.
- Note: In addition to updating LLA in-storage directories, CA PMO causes LLA to purge updated LNKLST modules from the VLF data space. These modules can reenter the data space in the future, depending upon program fetch activity for them.
- Reduced constraints on virtual storage. This is the case when you remove program load modules from LPA, and manage them with CA PMO and CA QuickFetch
- Increased knowledge of library directory search activity on your system. CA PMO provides you with statistics on library directory searches at the system, library, and member level. These statistics are presented online and as hardcopy reports.
- Simplified methods for managing library directories. CA PMO requires little training, and it is simple to operate.
- Management of the current LNKLST set as well as all previously current, active sets. CA PMO also automatically detects any new LNKLST sets activated by the SETPROG LNKLST ACTIVATE command.

CA PMO Features

Although CA PMO provides a wide variety of performance benefits, it is simple to install and to use, it does not require an IPL or system modifications, and it is invisible to end users. In addition to ease-of-use and convenience, CA PMO offers the following features:

- CA PMO automatically chooses all of the private libraries that it will handle. You may specify particular private libraries that CA PMO may or may not process, but this is optional.
- CA PMO automatically revises its managed list and creates new hash tables based on directory search activity. This allows CA PMO to enhance system performance for long periods of time without operator intervention.
- CA PMO automatically detects and handles updates to libraries. This feature makes it particularly useful for user-owned libraries, because users delete, modify, copy, and move the members of these libraries on a daily basis. CA PMO communicates information about updates across systems using optional components CA PMO Cross-System (PMO/XSYS) and PMO XCF Enhanced Data Transmission Component (PMO/XCF).
- CA PMO informs you of all of its activities. A monitor called PMOMON displays statistics on library directory searches on the system, along with savings of elapsed time, CPU time, and EXCPs. PMOMON also shows estimates of the monetary value of the benefits of CA PMO. It bases these estimates on the values that you assign to user time, CPU time, and EXCPs. PMOMON executes under TSO, CA Roscoe and as a batch program. If you want a library-by-library breakdown of directory search activity, CA PMO can generate a printed report that gives this information.
- CA PMO extends the usage of the storage hierarchy by placing PDS directories into processor storage. This helps to provide better response time in SMS and non-SMS environments.
- You can fine-tune the behavior of CA PMO by specifying various commands and parameters. For instance, you can designate groups of libraries that CA PMO must always handle, libraries that CA PMO may never handle, and you can determine baseline values that CA PMO uses when calculating various statistics. You can issue CA PMO commands at the operator console or in a batch job.
- CA PMO can automatically refresh LLA, or it can warn the operator that an LLA refresh is needed. LLA refreshes can be limited to the actual libraries or members that are updated. Alternatively, to eliminate the delays caused by LLA refreshes, you can let CA PMO manage all of your LNKLST libraries and discontinue use of LLA. CA PMO can issue selective LLA refreshes to libraries that are part of a dynamic LNKLST set, as well as to the IPL LNKLST set.

CA PMO and PDSE-type Libraries

CA PMO is unable to manage PDSE-type libraries due to PDSE structural characteristics. CA PMO now has a DESERV parameter that, when set, installs a Directory Entry Services exit that captures GET calls to intercept and resolve directory searches when a PDSE library is in the LNKLST or in a private library concatenation.

Although CA PMO does not manage the PDSE library, some updates to PDSE libraries within the LNKLST concatenation may result in the complete or selective flushing of entries that belong to PDS libraries that follow the updated PDSE library in the LNKLST. This is to ensure that an invalidated entry does not remain in the managed list after the PDSE update. For example, if the managed list contains an entry for a member of a PDS library later in the LNKLST concatenation than the PDSE library, then CA PMO must ensure that entry is flushed from the managed list if a member with the same name was added to the PDSE library.

With some types of updates to PDSE libraries, CA PMO does not receive information on the specific member or members that were updated. With these types of updates there is a complete flushing of managed list entries that belong to PDS libraries that follow the updated PDSE library in the LNKLST concatenation. Once flushed from the managed list, entries from the PDS libraries are again added as activity increases.

Note: In STEPLIB concatenation cases where the concatenation includes a PDSE and PDS libraries, CA PMO tries to resolve the directory search using the managed PDS libraries. However, the standard search sequence needs to be used. PMO may need to reissue the directory search for the PDSE libraries in the concatenation.

CA Auditor Support

CA PMO now supports the CA Auditor product. This enhancement adds data-only Program Description Modules (PDMs) that describe CA PMO front ends, SVC intercepts, and so on. It allows the auditing feature of CA Auditor to recognize CA PMO and keep it up to date with current development standards.

Migration and Upgrade Issues

The following issue should be kept in mind when migrating or upgrading from earlier releases of CA PMO:

- The PMO/XSYS control file needs to be formatted the first time CA PMO is started. This is due the control file header that differs from the header used in the previous releases.

Index