Tom Swift Revisits the Virtual Lookaside Facility

R.P. Shannon
Rocket Software
February 5, 2013
Session 12404

Rocket Software

Global software company

Approximately 1000 employees

Headquarters in Waltham, MA

**Offices in Atlanta, GA, Austin, TX,
Houston, TX, Chelyabinsk, RU, Dalian, China, etc.**

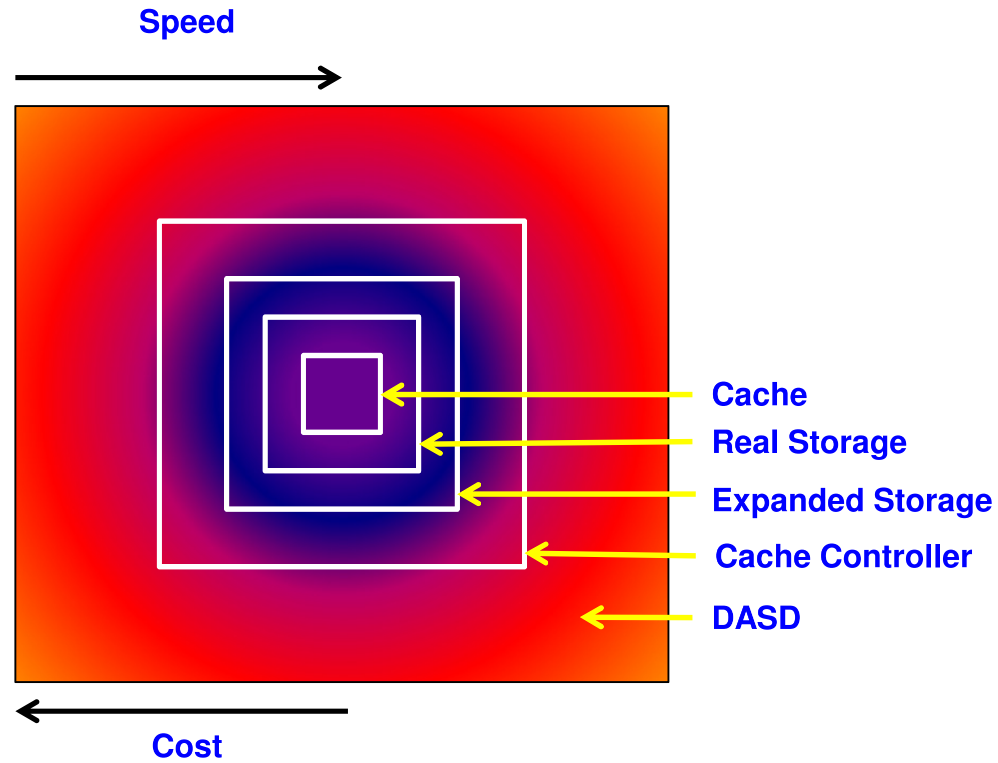
Produce many products that are branded and marketed by IBM

Former brands include Mainstar, Shadow, Bluezone and OpenTech

Tom Swift

Starring Wayne Morshhauser as Tom Swift

Storage Hierarchy



Normalized Access Times (1)

Cache: 1 second

Real Storage: 16 Seconds

Expanded Storage: 40 minutes

Cache Controller: 1 Day

DASD: 1 ½ weeks

Cache Access Times (2)

L1 cache - same cycle
L2 cache - 4 cycles
L3 cache in the same book - more than 100 cycles
L3 cache in another book - more than 200 cycles
Real Storage - about 850 cycles

(1) Courtesy of Wayne Morshhauser
(2) Courtesy of Greg Daynes

Additional Memory Types

2361 Large Capacity Storage (LCS) was an optional feature on the S/360 Models 50, 65 and 75:

- **Slower but cheaper than real storage**
- **Two region parameters (REGION=(X,Y)); one for regular memory, the other for LCS**

Expanded Storage:

- **Slower but cheaper than real storage**
- **Initially used for paging**
- **Different type of memory**
- **Accessed in 4K blocks**
- **Data must be moved from Estor to Cstor for processing**
- **Still used by zVM, but it is just real storage designated to be used as Estor**

System Z Flash:

- **Slower but cheaper than real storage**
- **Initially used for paging**
- **Uses flash drives**
- **I/O is performed to access the data, i.e., outside of the I/O boundary**

For more information on System Z Flash attend sessions 13057 and 13086 later today

Cache Concepts

"A cache is a place to hide things" Webster's Dictionary

Basic premise of caching in computer systems is the ability to re-read unchanged data

Candidates for caching:

- Should be frequently referenced
- Should have a high read/write ratio
- Provide the most benefit when accessed by multiple address spaces or systems

Everything that is cached must be backed by some type of storage

Data in caches tend to be volatile; it might not be there when needed

Searching and managing caches incurs some amount of overhead

Keep data as close to the processor cache as possible!

I/O Boundary

Denotes the place where data access switched from synchronous to asynchronous; occurs when referencing data outside of processor storage

Asynchronous retrieval:

- **Setup and schedule an I/O**
- **Save the state of the original task (Task A)**
- **Establish environment for a new task (Task B)**
- **Dispatch Task B**
- **Fill High Speed Buffer (HSB) with data for Task B**
- **Process interrupt for data retrieval for Task A**
- **Save state of Task B**
- **Re-establish environment for Task A**
- **Dispatch Task A**
- **Fill HSB with data for Task A**

I/O Elimination

In MVS/SP 3.1.0e (MVS/ESA) IBM discovered caching:

- **Data spaces and Hiperspaces provided additional data-only storage**
- **Data In Virtual (DIV) allowing ‘windowing” for linear data sets**
- **Virtual Lookaside Facility facilitated caching “objects”**

In MVS/SP 3.1.3 IBM introduced Hiperbatch (Data Lookaside Facility)

The only good I/O is a dead I/O

Full Speed Ahead

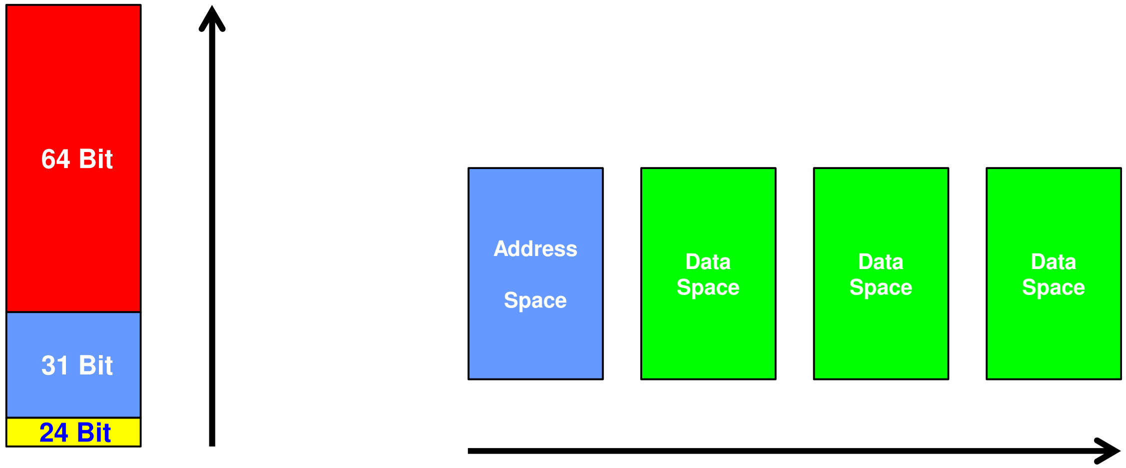
A computer always attempts to process at the highest speed possible

When you took an exam in college did you start with a 100 and work your way down, or did you start with a 0 and work your way up?

Computers start at 100

All of the multiple levels of caches, pipelining, out-of-order instruction execution etc., attempt to keep a computer running at 100

Vertical vs.. Horizontal Addressing Growth



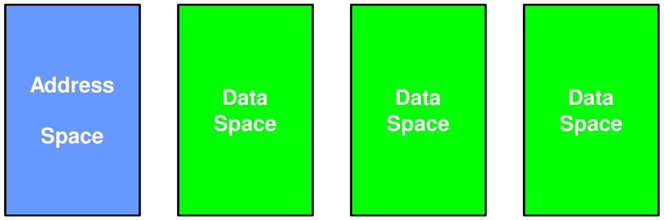
Vertical Growth:

- Requires significant architectural changes
- Implementation is slow, difficult, and expensive

Horizontal Growth:

- IBM encountered addressing limitations in MVS/XA
- Couldn’t implement bigger spaces (i.e., vertical growth) due to time and cost
- Instead implemented vertical growth, i.e., more spaces
- Two types of spaces: data spaces and Hiperspaces

Data Space Review



Data only spaces; maximum of 2 Gb; code cannot execute in a data space

Byte addressable; Hiperspaces are block (4K) addressable

No access to MVS common areas such as the Nucleus, CSA, PLPA or SQA

Access Registers contain the ALET for a data space; 32 bits not 64 bit

ALET is used to determine the Segment Table Origin

Must set the address mode to activate data space addressing; SAC 512

ALETs are unique affording data spaces some level of isolation

Each data space has a storage protect key (0-F)

A Common Area Data space (CADS) is automatically accessible by all address spaces, just like Common Storage

VLF Concepts

VLF is a caching facility that retains highly-used named objects in virtual storage to eliminate I/O operations

The objects are stored in data spaces that are owned and managed by VLF

Objects are byte-aligned in the data spaces and can be retrieved to byte boundaries in user storage

Objects reside in pageable storage

VLF may trim, i.e., delete, objects from the data spaces

Callers must be able to refetch, reload or recreate objects

An application may terminate a class due to errors

VLF may terminate a class due to various errors

VLF Initialization

VLF runs as a non-swappable started task

Service Class SYSSTC

Put the start command in COMMNDxx (because it usually executes before automation products)

COM='S VLF,SUB=MSTR,*nn=xx*'

- SUB-MSTR is required so that VLF can start before JES
- nn=xx specifies the COFVLF suffix and is only required when the suffix does not equal 00

VLF Procedure:

```
//VLF      PROC  NN=00
//VLF      EXEC  PGM=COFMINIT,PARM='NN=&NN'
```

If VLF starts before the VLF users, the users will begin using VLF once it has been initialized

VLF can be stopped by issuing a "P VLF" command; stopping VLF will degrade system performance

COFVLFxx Parmlib Member

```
CLASS NAME(CSVLLA)
      EMAJ(LLA)
      MAXVIRT(8096)
CLASS NAME(IKJEXEC)
      EDSN(RS21.LIBDEF.EXEC)
      EDSN(RSPLEX01.LIBDEF.EXEC)
      EDSN(ISP.SISPCLIB)
      MAXVIRT(512)
CLASS NAME(IGGCAS)
      EMAJ(CATALOG.RSPLEX01.OMGR.CAT1)
      EMAJ(CATALOG.RSPLEX01.USERCAT)
      EMAJ(ICF.RSPLEX01.DB2.CAT1)
      EMAJ(ICF.RSPLEX01.IMS.A3DB.CAT1)
      MAXVIRT(2048)
CLASS NAME(IRRGTS)
      EMAJ(GTS)
CLASS NAME(IRRACEE)
      EMAJ(ACEE)
CLASS NAME(IRRGMAP)
      EMAJ(GMAP)
CLASS NAME(IRRUMAP)
      EMAJ(UMAP)
CLASS NAME(IRRSMAP)
      EMAJ(SMAP)
```

ALERTAGE parameter specifies the age of an object in seconds, used by Health Check IBMVLF,VLF_MAXVIRT to determine if trim occurs too frequently. Default = 60

Terminology

Class: a set of related objects; Example: IKJEXEC is a class used by TSO

Major Name: a group within a class; Example: SYS3.CLIB (a clist library)

Minor Name: a specific object within a major name; Example: #ISMF (a clist)

Within a class, each major name must be unique; within a major name, each minor name must be unique

Hashed Object Name = Class|Major|Minor

MAXVIRT: The maximum amount of data space storage for objects; the default is 4096 4K blocks (16 Mb)

Trim: VLF begins culling objects it has used about 90% of the MAXVIRT value

VLF Data Spaces

VLF creates two data spaces per class when COFIDENT is issued:

“Data” data space contains the objects:

- **Size controlled by MAXVIRT parameter**
- **Name is D+classname**

Example: DCSVLLA for LLA

“Control” data space:

- **Size is 2 Gb (but usually only a small amount is used)**
- **Name is C+classname**
- **Contains control structures such as:**
  - **Pointers to the objects**
  - **Size of the objects**
  - **And more ….**

Example: CCSVLLA for LLA

VLF Dataspaces (D J,VLF)

```
VLF      VLF      VLF      NSW  S   A=001F   PER=NO   SMC=000 PGN=N/A
                              DMN=N/A  AFF=NONE
                              CT=000.274S  ET=20.02.29
                              WKL=SYSTEM   SCL=SYSSTC   P=1
                              RGP=N/A      SRVR=NO  QSC=NO
                              ADDR SPACE ASTE=0BDC27C0
                              DSPNAME=DIKJEXEC ASTE=0B628600
                              DSPNAME=CIKJEXEC ASTE=076ECF80
                              DSPNAME=DIRRGMAP ASTE=0B628300
                              DSPNAME=CIRRGMAP ASTE=0A67DA00
                              DSPNAME=DIRRUMAP ASTE=0B628280
                              DSPNAME=CIRRUMAP ASTE=0A67D980
                              DSPNAME=DIRRSMAP ASTE=0B628200
                              DSPNAME=CIRRSMAP ASTE=0A67D900
                              DSPNAME=DCSVLLA  ASTE=0B628180
                              DSPNAME=CCSVLLA  ASTE=0A67D580
                              DSPNAME=DIRRACEE ASTE=0B628100
                              DSPNAME=CIRRACEE ASTE=0A67D480
                              DSPNAME=DIGGCAS  ASTE=0B628080
                              DSPNAME=CIGGCAS  ASTE=7E9CB580
```

Data spaces that begin with "D" contain objects
Data spaces that begin with "C" contain control information

VLF Services

COFDEFIN	Define a class
COFIDENT	Identify a user
COFCREAT	Create an Object
COFRETRI	Retrieve an object
COREMOV	Remove a user
COFPURGE	Purge a class
COFNOTIF	Indicate Change

COFDEFIN: Define a Class

COFDEFIN

CLASS	**Class**
MAJLEN=*majlen*	**Major Length; 1-64; PDS is always 50**
MINLEN=*minlen*	**Minor Length; 1-64; for PDS is always 8**
,TRIM=<u>ON</u> | OFF	**Permit Trim?**
,AUTHRET=<u>NO</u> | YES	**Authorized Retriever? Supervisor state or key 0-7**
,RETCODE=*retcode*	**Return Code**
,RSNCODE=*rsncode*	**Reason Code**

Note: Each class *must* have an entry in Parmlib
COFDEFIN issued once per class
Data spaces are created when the class is defined

COFIDENT: Connect a Caller to a Class

COFIDENT	
DDNAME=*ddname*	**DDname**
MAJNLST=majnlst	**Major Name**
,CLASS=*class*	**Class; 7 character name from COFDEFIN**
,SCOPE=<u>HOME</u> | SYSTEM	**Scope of services that can retrieve objects**
,UTOKEN=utoken	**16 byte token returned by VLF**
,RETCODE=*retcode*	**Return Code**
,RSNCODE=*rsncode*	**Reason Code**

COFCREAT: Create an Object

COFCREAT

MAJOR-major	**Use for non-PDS only**
CINDEX=cindex	**Concatenation index; required for PDS class**
DDNAME=ddname	**Ddname; required for PDS**
,REPLACE = NO | YES	**Replace existing object?**
MINOR=minor	**Minor name**
UTOKEN=utoken	**16 byte token from COFIDENT**
OBJPRTL=objprtl	**Object parts list; ALET, Part addr, Part length**
OBJPLSZ-objplsz	**Size of parts list**
,RETCODE=*retcode*	**Return Code**
,RSNCODE=*rsncode*	**Reason Code**

Note: Normal processing is to attempt to Retrieve an object, and if unsuccessful, obtain the object from the permanent source and then issue COFCREAT

COFRETRI: Retrieve an Object

COFRETRI

MINOR=*minor*	**Minor name of the object**
UTOKEN=*utoken*	**Token from COFIDENT**
TLIST=*tlist*	**Target Area List; ALET, Target Addr, Target size**
TLSIZE=*tlsize*	**Target area List Size**
OBJSIZE=*objsize*	**Object size; returned by VLF**
CINDEX=*cindex*	**Concatenation index**
,RETCODE=*retcode*	**Return Code**
,RSNCODE=*rsncode*	**Reason Code**

COFREMOV: Remove (Disconnect) a User

COFREMOV	
UTOKEN=*utoken*	**Token from COFIDENT**
,RETCODE=*retcode*	**Return Code**
,RSNCODE=*rsncode*	**Reason Code**

COFPURGE: Purge (Delete) a Class

COFPURGE	
CLASS=*class*	**Class**
,RETCODE=*retcode*	**Return Code**
,RSNCODE=*rsncode*	**Reason Code**

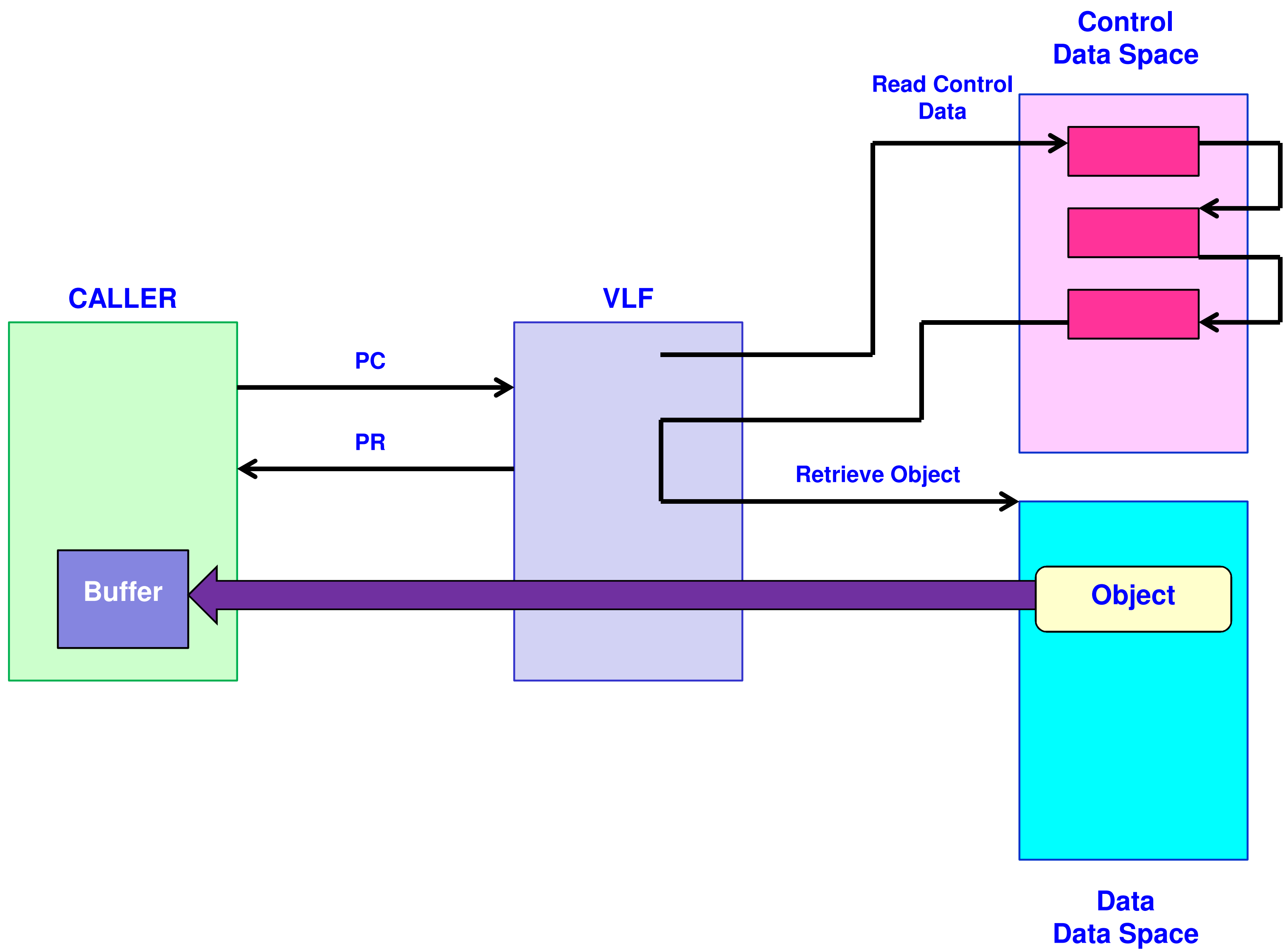
COFNOTIF: Notify VLF of Changes

COFNOTIF

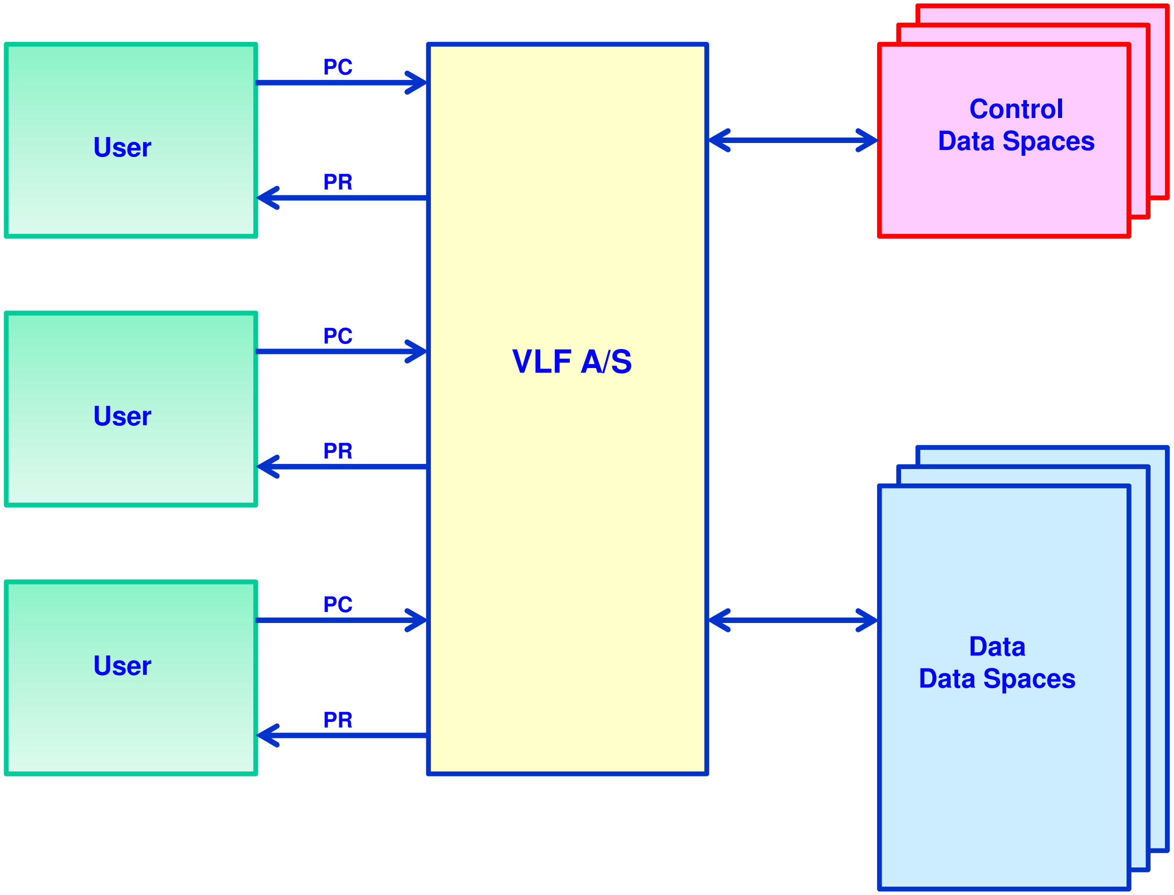
FUNC=DELMAJOR | DELMINOR | ADDMINOR | UPDMINOR | PURGEVOL

,MAJLIST=*majlist*	**Required for FUNC=DELMAJOR**
,MAJNUM=*majnum*	
.MAJLEN=*majlen*	
,MAJOR=*major*	
,MINLIST=*minlist*	**Reqd for FUNC=DELMINOR, ADDMINOR, UPDMINOR**
,MINLEN=*minlen*	
,VOLUME=*volume*	
,CLASS=*class*	
,RETCODE=*retcode*	**Return Code**
,RSNCODE=*rsncode*	**Reason Code**

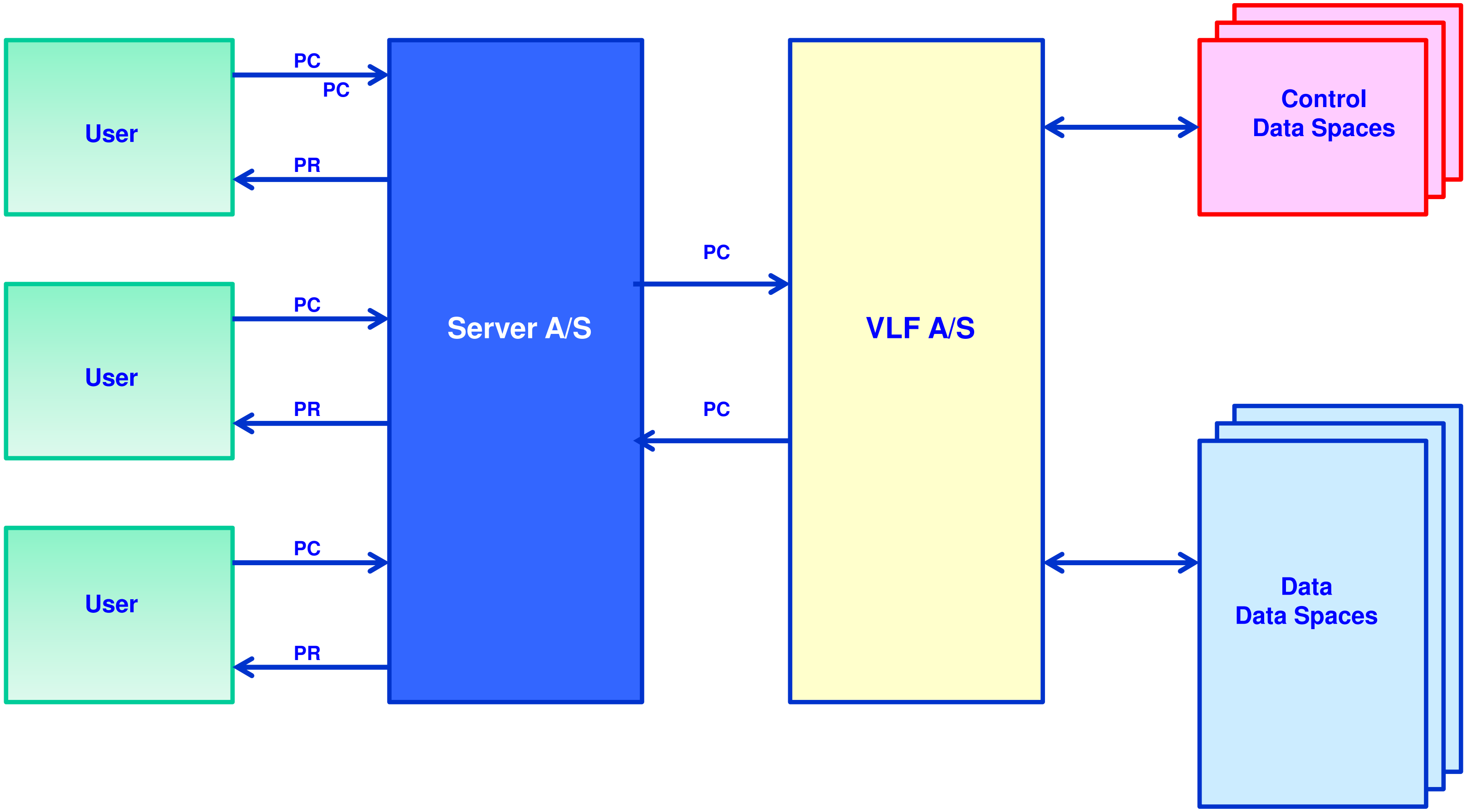
Object Retrieval



VLF Application Without A Server A/S



VLF Application With A Server A/S



LLA Use of VLF

LLA can manage Linklist (LNKLSTxx) and non-Linklist (CSVLLAxx) libraries

Libraries can be frozen or non-frozen; Linklist libraries are frozen by default

For frozen libraries the LLA directory is used; built during LLA initialization

For non-frozen libraries the directories on DASD are used; I/O is required for each directory search

Frozen libraries provide *much* better performance than non-frozen

LLA will cache modules in VLF for both Linklist and non-Linklist libraries, for frozen and non-frozen libraries

To determines which modules to cache in VLF:
- **LLA maintains statistics on all fetches**
- **After 2000 fetches from a library or 10 fetches of a module, module staging analysis is performed**
- **CSVLLIX2 can be used to influence staging**

LLA will only attempt to retrieve from VLF the objects it has already cached

LLA REFRESH vs.. UPDATE

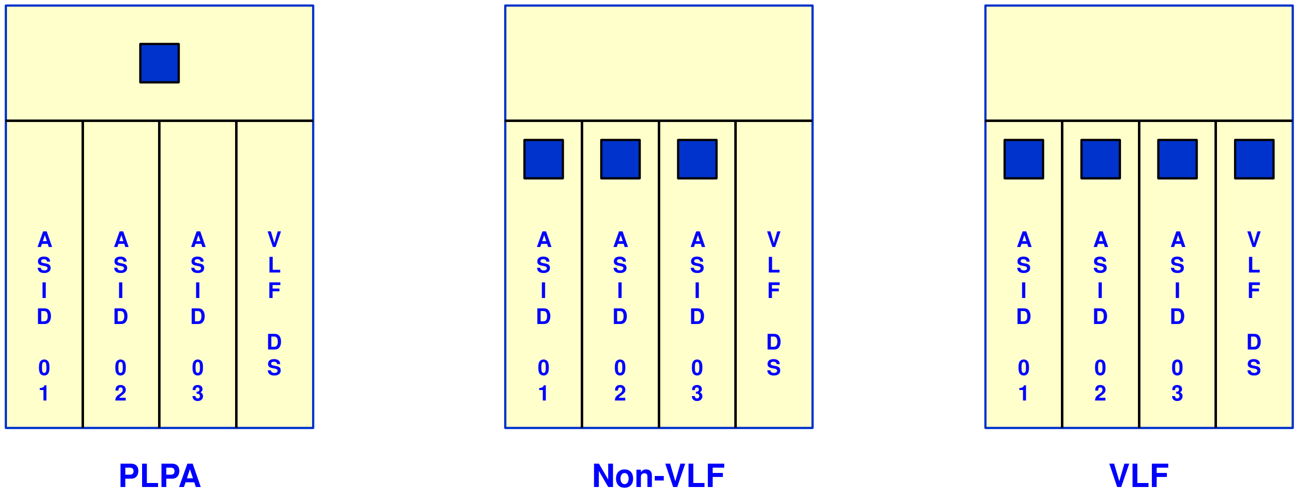
Modify LLA,REFRESH:

- **Rebuilds the entire LLA directory**
- **Flushes VLF**
- **Easy command to issue, but severe performance degradation can occur**

Modify LLA,UPDATE=xx

- **xx indicates a CSVLLAxx member that contains control statements**
- **Kind of a pain to issue the command it requires knowledge of what is being changed and requires some set up**
- **Selectively refreshes whatever is specified; much less disruptive than REFRESH**

PLPA vs.. VLF vs.. Fetch Storage Utilization



PLPA: One copy for the entire system. Modules must be reenterable. No I/O after PLPA is built.

Non-VLF: A unique copy is fetched for each address space that requires it.

VLF: First requester fetches module and caches it in VLF. Subsequent requests by other address spaces are satisfied from VLF instead of fetching the module again (I/O elimination).

Access from VLF vs. Fetch

Module	Alias	Length	Fetch	Duration	Jobname	ASID	#LLAF	#PGMF
IDCCDAL		00005618	PGM	00.000372	CQMDWG4	024B	3088	150
IDCCDDE		0000DAA0	LLA	00.000038	R91BDBM1	0134	10182	115
IDCCDDL		00000A50	LLA	00.000012	TG23866B	004F	10846	161
IDCCDLC		00000B80	LLA	00.000008	MDDECRB	024A	1179	105
IDCCDPM		000008C8	LLA	00.000009	EMCSCF	0110	21593	285
IDCCDPR		00001208	PGM	00.000798	GGC#LINK	0042	40	11
IDCCDRP		00001ED0	PGM	00.000537	COPYPROF	004F	30	51
IDCCDTC		00000018	PGM	00.000292	SMFTEST	0052	0	10
IDCCDVY		00000158	LLA	00.000007	SMFDUMP	0237	95	75
IDCDE01		0000F7B0	PGM	00.001224	R91BDBM1	0134	9724	573
IDCDL01		00009000	PGM	00.000721	TG23866B	004F	10370	637
IDCIO04		00000480	LLA	00.000008	MDDECRB	024A	1842	302
IDCLA01	IDCSS01	000108D8	PGM	00.001217	S3TMS02	01E8	1	10
IDCLC01		0003FC40	PGM	00.003742	MDDECRB	024A	715	569
IDCPM01		00000CB8	LLA	00.000010	EMCSCF	0110	21679	199

TSO Use of VLF

Only libraries concatenated to the SYSPROC DD statement are supported by VLF; libraries concatenated to SYSEXEC are *not* supported

The SYSEXEC concatenation is searched before the SYSPROC concatenation; (when VLF was introduced SYSPROC was searched first)

Clist processing:

- Phase 1: Read the Clist, build the in-storage procedure, and put the procedure on the command stack
- Phase 2: Removes and executes each statement from the stack
- Clists are cached in VLF after Phase 1

Rexx Programs:

- Fetch the Rexx program
- Execute the Rexx program (include interpretation)
- Rexx programs are stored in VLF after fetch

VLF potentially provides more benefits for Clist processing, but Rexx programs will still benefit from I/O avoidance

VLF combined with the Rexx Compiler can provide lots of benefits to Rexx processing

TSO VLF Eligibility

Eligibility for Implicit Execution

Level	DDname	Type
SYSTEM	SYSEXEC	Rexx
	SYSPROC	Clist or Rexx
APPLICATION	Any Name	Rexx
	Any Name	Clist or Rexx
USER	SYSUEXEC	Rexx
	SYSUPROC	Clist or Rexx

Rexx programs that reside in libraries concatenated to any of the "SYSPROC" DDnames must have /* REXX */ coded in the first line to identify it as a Rexx program instead of a Clist

Explicitly executed Clists and Rexx Programs are not eligible for VLF processing as they are not associated with a DDname

TSO VLF Effectiveness Factors

Libraries concatenated to SYSEXEC cannot be managed

Large Clists with high Phase 1 processing will benefit the most from VLF caching

Although VLF provides more benefits to Clist processing, Rexx programs will still benefit from eliminating I/O

VLF combined with the Rexx Compiler can provide lots of benefits to Rexx processing

Put your Clists and Rexx into one library (or a few libraries) and define that library to VLF

ISP.ISPCLIB is good candidate for VLF

Rexx programs/Clists used to trigger dialogs are good candidates

Rexx programs/Clists that are changed frequently should not be VLF-managed

Not VLF-specific, but the managed Rexx programs/Clists should be named so that they are found first in the concatenation

The Agony Of VLFNOTE

VLFNOTE needs to be in the IKJTSOxx AUTHCMD list; access should be protected by a security product

Changes to objects loaded from EDSN must be communicated to VLF

Of the standard IBM VLF classes, IKJTSO is the only one that uses EDSN

Since changes aren't automatically detected, if VLFNOTE isn't issued VLF will continue to use the unchanged object; this can be very frustrating as the person who changed the object won't understand why the changes aren't recognized

This may be a deterrent to using the IKJTSO class

Catalog Use of VLF

CAS (Catalog Address Space) stores a record (object) in a VLF data space whenever a record is read by key

VLF caching is sometimes called Catalog Data Space Cache (CDSC)

Master Catalog (MCAT) records are cached in the Catalog Address Space so don't define the Master Catalog to VLF

Catalog updates are maintained in the VVDS:

- When the system accesses a catalog, it reads the VVDS and deletes the changed entries from VLF
- The updates wrap after 92 entries and the update history is lost causing the VLF catalog objects to be purged
- So don't define high activity catalogs to VLF on systems with low activity; insure activity is relatively balanced

Catalog Modify commands:

MODIFY CATALOG,VLF|NOVLF(catname)	add/remove a catalog to VLF
MODIFY CATALOG,NOVLF(catname)	remove a catalog from VLF
MODIFY CATALOG,OPEN	show catalogs that are open
MODIFY CATALOG,REPORT,CACHE	report on catalogs using VLF
MODIFY CATALOG,ALLOCATED	report on allocated catalogs

Note: Attend sessions 12977 & 12978 tomorrow to learn about forthcoming CAS changes

MODIFY CATALOG,ALLOCATED

```
*CAS**************************************************
* YSV-E- OMP100 0001 CATALOG.RSPLEX01.OMGR.CAT1                    15
*
* YSI-E- IMP100 0001 ICF.RSPLEX01.IMS.CAT1                          1
*
* YSI-R- S3P100 0001 ICF.RSPLEX01.SHARED.CAT1                       1
*
* YSV-R- QXP101 0001 ICF.RSPLEX01.QBX2.CAT                          1
*
* YSV-R- S1P10B 0001 CATALOG.RSPLEX01.USERCAT                      50
*
* YSV-R- QXP101 0001 ICF.RSPLEX01.QBX2.CAT                          1
*
* YSV-E- DVP101 0001 ICF.RSPLEX01.DEV.CAT1                         55
*
* YSI-R- R3P100 0001 CATALOG.RSRTE.CAT1                             1
*
* Y-I-E- MCP100 0001 CATALOG.RSPLEX01.MASTER                        1
*
*******************************************************************
*
* Y/N-ALLOCATED TO CAS, S-SMS, V-VLF, I-ISC, C-CLOSED, D-DELETED,
*
* R-SHARED, A-ATL, E-ECS SHARED, K-LOCKED
*
*CAS***************************************************************
```

MODIFY CATALOG,REPORT,PERFORMANCE

```
*CAS*************************************************
*  Statistics since 17:11:00.38 on  01/05/2013      *
*  -----CATALOG EVENT----   --COUNT--  ---AVERAGE---  *
*  Entries to Catalog           2,889K     3.234 MSEC  *
*  BCS ENQ Shr Sys              5,570K     0.145 MSEC  *
*  BCS ENQ Excl Sys            79,099      0.615 MSEC  *
*  BCS DEQ                      6,886K     0.024 MSEC  *
<snip>
*  VLF Delete Major               216      0.032 MSEC  *
*  VLF Delete User                  1      0.003 MSEC  *
*  VLF Create Minor           278,099      0.008 MSEC  *
*  VLF Retrieve Minor           2,931K     0.003 MSEC  *
*  VLF Delete Minor           131,485      0.009 MSEC  *
*  VLF Define Major                 1      0.152 MSEC  *
*  VLF Identify                 1,746      0.003 MSEC  *
*  RMM Tape Exit               10,039      0.000 MSEC  *
*  OEM Tape Exit               10,039      0.000 MSEC  *
*  BCS Allocate                   157      8.382 MSEC  *
*  BCS Deallocate                   6      3.525 MSEC  *
*  SMF Write                  344,344      0.046 MSEC  *
*  ENQ SYSZCATS Shr                15      0.046 MSEC  *
*  IXLCONN                          2     81.485 MSEC  *
*  IXLCACHE Read                4,974K     0.006 MSEC  *
*  IXLCACHE Write             242,696      0.005 MSEC  *
*  Resolve Symbolic             2,516      0.025 MSEC  *
*  MVS Allocate                   146      8.932 MSEC  *
<snip>
```

RACF Use of VLF

RACF uses VLF to cache ACEEs (Accessor Environment Element)

An ACEE is cached for each address space:

- **If you have three TSO sessions there will be three cached ACEEs**
- **If you run a batch job there will be one ACEE**
- **So the same ACEE may be cached multiple times**

Before activating the IRRACEE class check for the use of ACEEICE field:

- **Pointer to user-defined data**
- **Exits IRRACX01 and IRRACX02 are used to tell RACF what to do with the user data**
- **VLF may cause problems if ACEEICE and the aforementioned exits are used**
- **Read the documentation**

For security-related changes where all of the incorrect user ACEE entries cannot be determined, all the ACEEs will be removed from VLF

Commands that make security-related changes include ALTUSER, DELUSER, and ADDUSER

The more groups a user is connected to the greater the size of the object cached in VLF

Other than the ACEEICE, don't worry about IRACEE too much; it works - use it

RACF Use of VLF – The “Oddball” Classes

RACF created a number of VLF classes to exploit DIM:

- **IRRGMAP: contains mapping of GIDs to a Group Names**
- **IRRUMAP: contains mappings of UIDs to User Ids**
- **IRRSMAP: contains User Security Packets (USPs) for thread level security**
- **IRRSPS0: contains Signature Verification Data for signed programs**

These classes have very low activity on most systems

But defining them won’t hurt anything and may result in slight performance gains

Diagnostics

Component trace:

TRACE CT,ON,COMP=SYSVLF
TRACE CT,OFF,COMP=SYSVLF

Process the component trace data with IPCS:

CTRACE COMP(SYSVLF) FULL | SHORT |SUMMARY

To dump VLF and the data space(s):

DUMP COMM=‘Dump of VLF”
R XX,JOBNAME=VLF,CONT
R YY,DSPNAME=(VLF.DCVSLLA,VLF,CCSVLLA),END

IPCS browse function:

PTR	Address	Address Space	
0001	00000000	ASID(X'001F')	
0002	00000000	ASID(X'001F")	DSPNAME(DCSVLLA)
0003	00000000	ASID(X'001F')	DSPNAME(CCSVLLA)

Note: On my system the VLF address space is X’001F’

SMF Type 41 Subtype 3

One record produced every 15 minutes

Each record contains all of the VLF classes

Record contains:

- **Class name**
- **Maximum virtual storage (in 4K blocks)**
- **Amount of storage used (in 4K blocks)**
- **Number of times searched**
- **Number of objects found**
- **Number of objects added**
- **Number of objects deleted**
- **Number of objects trimmed**
- **Size of the largest object attempted to add to the cache**

It would be very helpful to identify the largest object attempted to add to the cache

Sample SMF Reports

Class	Maxvirt	Maxused	Searches	Found	Del	Trim	Largest
CSVLLA	65,536	18,132	540,820	540,820	0	0	9,404,632
IRRACEE	12,288	2,976	95,150	87,407	60	0	167,960
IRRGTS	256	2	0	0	0	0	20
IRRUMAP	4,096	1	997	742	0	0	8
IRRSMAP	4,096	1	44	29	0	0	162
IRRGMAP	4,096	1	252	51	0	0	8
IGGCAS	2,560	2,325	201,034	136,721	7,351	15,671	23,030
IGGCAS	2,048	1,764	215,765	58,026	3,755	124,001	16,626
IKJEXEC	800	345	641	627	0	0	103,784

The hit ratio for CSVLLA is misleading because LLA will only search VLF for objects it previously cached; the CSVLLAX1 fetch exit can provide accurate statistics

540,820 searches in 15 minutes is > 600 per second!

The second IGGCAS example shows very high trim activity which indicates MAXVIRT is too low

Environmental Changes Since VLF Was Introduced

RAID vs. SLED

Ficon/Escon vs. Bus & Tag

zHPF

Huge amounts of DASD cache

PDS Search Assist

Vast amounts of real storage

Rexx has overtaken Clist

BLOATware

But VLF continues to be viable, particularly for LLA and RACF

Contributors

I'd like to acknowledge the following contributors to this session:

Sam Knutson, Gieco

Brian Scott, Data-Tronics

Bibliography

Security Server RACF System Programmer's Guide; SA23-2287; IBM Corporation

DFSMS Managing Catalogs; SC26-7409; IBM Corporation

MVS Installation Exits; SA22-7593; IBM Corporation

MVS Initialization and Tuning Reference; SA22-7592; IBM Corporation

Authorized Assembler Services Reference Volume 1 (ALE-DYN); SA22-7609; IBM Corporation

TSO/E Customization; SA22-7783; IBM Corporation

United States Patent; Virtual Lookaside Facility; Application Number: 07/225,445; Morshhauser et al.

MVS/SP 3.1.0 Virtual Lookaside Facility; SHARE 72, Session 0315; W. J. Morshhauser