VSAM record-level data sharing

by J. P. Strickland

VSAM (Virtual Storage Access Method) has been extended to provide multisystem data sharing through use of the S/390® coupling facility. This paper describes the serialization techniques used in the implementation of VSAM data sharing.

This paper begins with a brief history and an overview of the VSAM (Virtual Storage Access Method) data architecture. Then a description of the new VSAM record-level sharing function is given. This coupling-facility-based data sharing is an integral part of the System/390* (S/390*) Parallel Sysplex*.

History of VSAM

Prior to VSAM, Operating System/360 (OS/360) provided a number of data access methods with a variety of data formats and organizations. Examples are: BSAM (Basic Sequential Access Method), QSAM (Queued Sequential Access Method), BDAM (Basic Direct Access Method), BISAM (Basic Indexed Sequential Access Method), and QISAM (Queued Index Sequential Access Method). One objective of VSAM was to provide a single data format and organization and access functions for data stored on DASD (direct access storage device).

The first release of VSAM was shipped in 1973. It was introduced with System/370*.

At the time VSAM was being defined, System/360* DASD architecture introduced a function called *RPS* (rotational position sensing). This function was a significant advancement because it provided a basis for concurrently executing multiple DASD I/O channel programs on a single I/O channel. The idea is that, while a device is busy positioning itself to access a requested record, its control unit can disconnect from the channel, allowing the channel to execute another channel program. When the device is in position near the record, it requests reconnection to the channel and then sends or receives the record across the channel. Neither the BISAM or QISAM key search function could exploit the RPS function. But, the keyed data format of VSAM with its compressed index is able to exploit the RPS function.

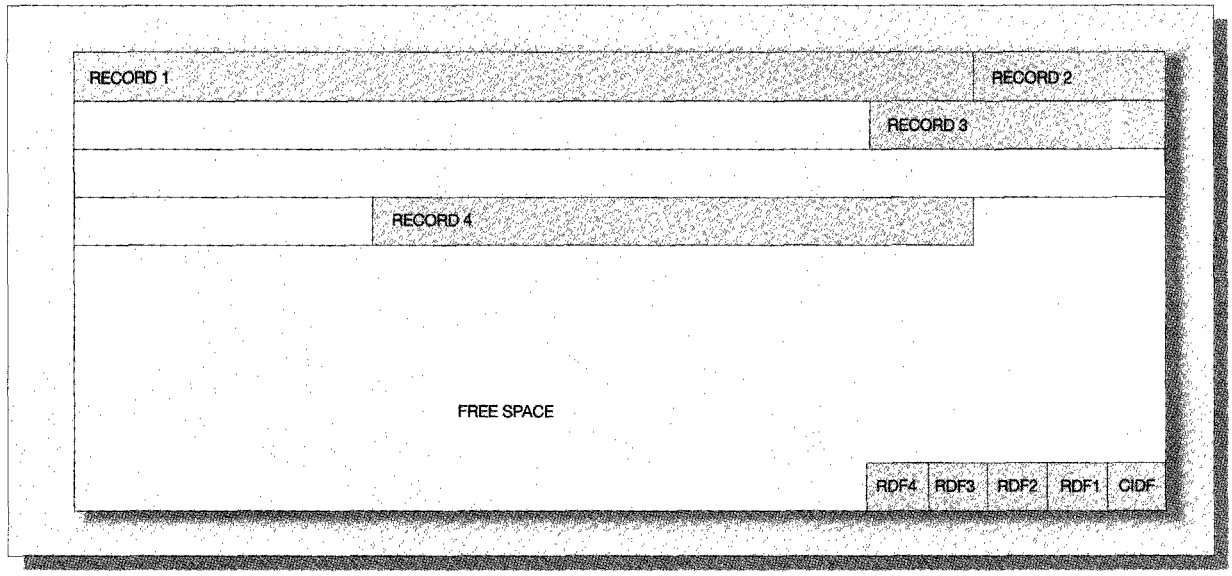
CICS* (Customer Information Control System) is a widely used transaction processing system. CICS provides a file access interface on top of VSAM. It is a CICS file control function that includes transactional recovery for VSAM files. This isolation and rollback capability enables VSAM data to be shared among CICS applications.

VSAM architecture

The data format of VSAM supports fixed-length and variable-length records. The records are mapped into fixed-length DASD read and write units called *control intervals* (CIs).

Control information resides at the end of each control interval. There are two sets of control information. The last four bytes of the CI is the control interval definition field (CIDF). Immediately to the left of the CIDF is one or more record definition fields (RDFs). An RDF describes one record, a set of con-

Figure 1 Layout of a VSAM data control interval



tiguous fixed-length records, or a segment of a spanned record. A spanned record has a length greater than the size of a control interval, and thus it resides in a set of control intervals.

The control interval architecture includes a distributed free space capability. Records are stored within a control interval starting at offset zero. The RDFs specify the lengths of each record and are stored in reverse order beginning immediately to the left of the CIDF. Any unused space between the end of the last record in the CI and the last RDF is free space. The format of a VSAM data control interval is shown in Figure 1.

The basic VSAM data addressing model is a linear space model. The term RBA (relative byte address) is used to designate the address of a data location within a VSAM linear space. Although the byte address technique is used, the VSAM data are stored and accessed as records.

VSAM data set types. VSAM provides a number of data set types or data organization schemes. They are:

- Key-sequenced data set (KSDS)
- Entry-sequenced data set (ESDS)
- Relative record data set (RRDS)
- Variable-length relative record data set (VRRDS)
- Linear data set (LDS)

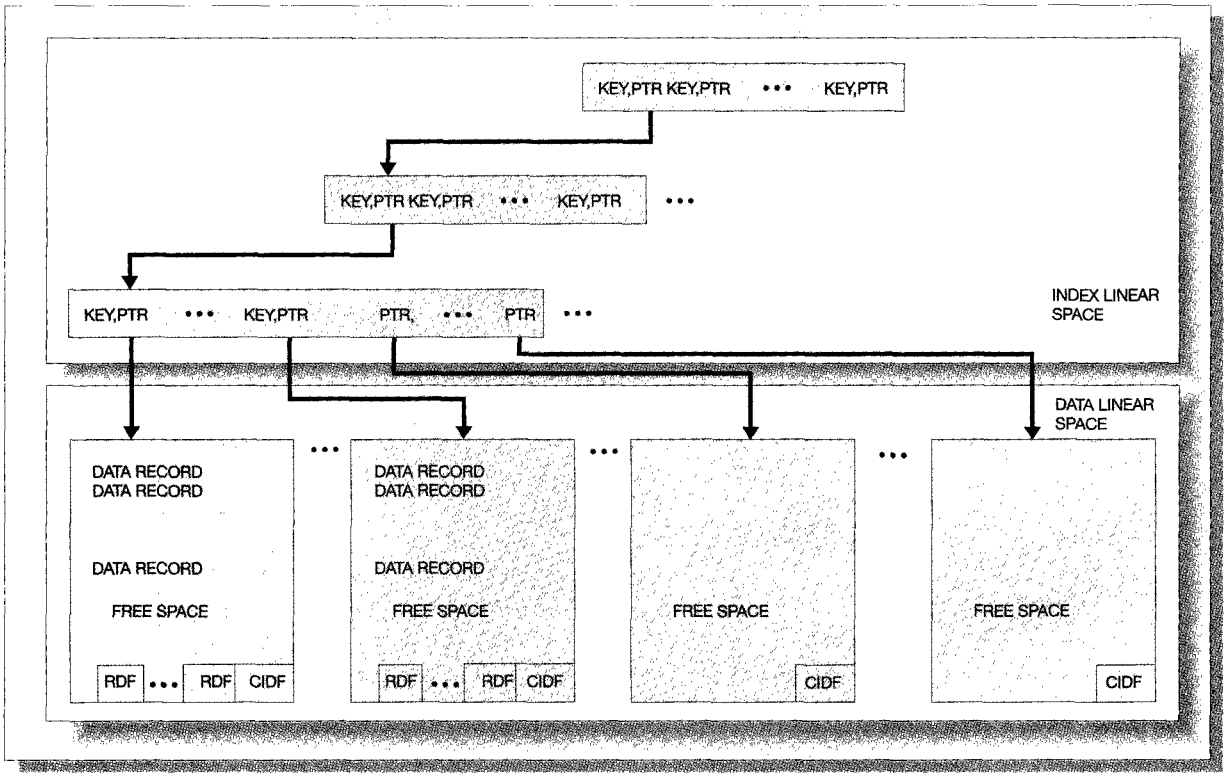
VSAM key-sequenced data set architecture. The key-sequenced data set (KSDS) is a widely used VSAM data set type. Details of its data and index architecture are described in this subsection. Details of the other VSAM data set types will not be described in this paper.

A KSDS consists of two linear spaces. The data records are stored in one linear space, and the index records are stored in a second linear space.

Each record of a KSDS contains a key field. The offset within the record to the beginning of the key field and the key length are user-defined. VSAM does not support variable-length keys within a single KSDS. For a specific KSDS, the key length is fixed at a value ranging from 1 to 255 bytes. The key field is treated as a binary value. The data records are stored within a control interval in ascending key sequence. As records are updated to change their length, new records are inserted, records are erased, and any higher-key-value records within the data control interval are shifted to maintain key sequence.

In addition to control intervals, the VSAM KSDS architecture defines control areas. A control area (CA)

Figure 2 KSDS index and data structure



is a set of contiguous control intervals. This area defines a two-level space hierarchy within the linear space of the data component of the KSDS. For each control area of the data component, VSAM assigns a control interval of the index component's linear space. This index control interval is called a *sequence set control interval.* The sequence set control interval contains one entry for each data control interval within the corresponding data control area.

A data control interval may contain data records, or it may be empty. An empty data control interval is called a *free space control interval.* The index sequence set control interval of the data control area contains two sets of information. One set is a key-ordered index containing a compressed key value and a data control interval pointer for each data control interval that contains data records. The other set is a list of free space data control intervals. The compressed key value in a sequence set entry represents the highest data record key that may be stored in the corresponding data control interval.

In order to reduce index search time, a multilevel index is maintained. The index level immediately above the sequence set contains one index entry for each sequence set control interval. The entries are maintained in ascending key sequence. An index entry is not allowed to span an index control interval. Space is allocated within the index at the granularity of an index control interval. An index tree is maintained where the top level of the index tree consists of a single index control interval. The entries in this level of the index point down to index control intervals in the next lower lever of the index, continuing to the lowest level of the index.

Figure 2 shows the relationship between the VSAM KSDS index and data linear spaces. It illustrates a KSDS with a three-level index. Only the first data control area is shown. The data records are stored within data control intervals. A sequence set index control interval exists for each data control area. It shows the key order of the data control intervals within the data control area and the set of free space control

intervals within this data control area (if any). The index above the sequence set forms a tree structure. The records within each data control interval are maintained in key sequence.

The VSAM KSDS architecture provides distributed free space. Free space is maintained within each data control interval and a set of empty data control intervals within each control area. As records are inserted, the position where they are stored is determined by the index and the existing records within the data set. When a record is inserted within a data control interval, any existing data records within that control interval with higher key values are shifted to make room for the new record.

The heart of the S/390 Parallel Sysplex is its data-sharing technology based on the coupling facility.

When an insert occurs without sufficient free space within the data control interval, the data control interval is split. The split consists of allocating a free space control interval from the free list within the sequence set control interval of the control area, and distributing records across the two data control intervals and inserting the record. When there is not a free space data control interval within the data control area, a control area split is performed. This split allocates a new control area from the end of the data set and distributes the data across the two control areas, forming free space control intervals within each of the two control areas. When a control interval or control area split occurs, the index is updated to reflect the split.

When a record is deleted from a data control interval, any existing higher keyed records within the control interval are shifted, causing the free space beyond the highest keyed record to grow. When the last data record is erased from a data control interval, its sequence set entry is removed from the index, and the data control interval pointer is added to the free data control interval list of the sequence set.

VSAM alternate indexes

In addition to the primary index of a KSDS, VSAM supports alternate indexes. Alternate indexes may be defined over KSDSs or ESDSs. The KSDS or ESDS is called the *base cluster*. In order to have an alternate index, each record in the base cluster must contain an alternate key field. The key field is at a fixed offset within each record and has a fixed length ranging from 1 to 255 bytes. VSAM implements the alternate index as an internal KSDS, where a record in this KSDS consists of the alternate index key from a record in the base cluster and the record identifier of the base cluster record. For a KSDS base cluster, the record identifier is the primary key of the record. For an ESDS, it is the RBA of the record. Multiple alternate indexes may be defined over a base cluster.

VSAM supports concurrent access to the base cluster records via the base cluster and the alternate indexes. VSAM provides an option to immediately upgrade the alternate indexes when changes are made to the base cluster records.

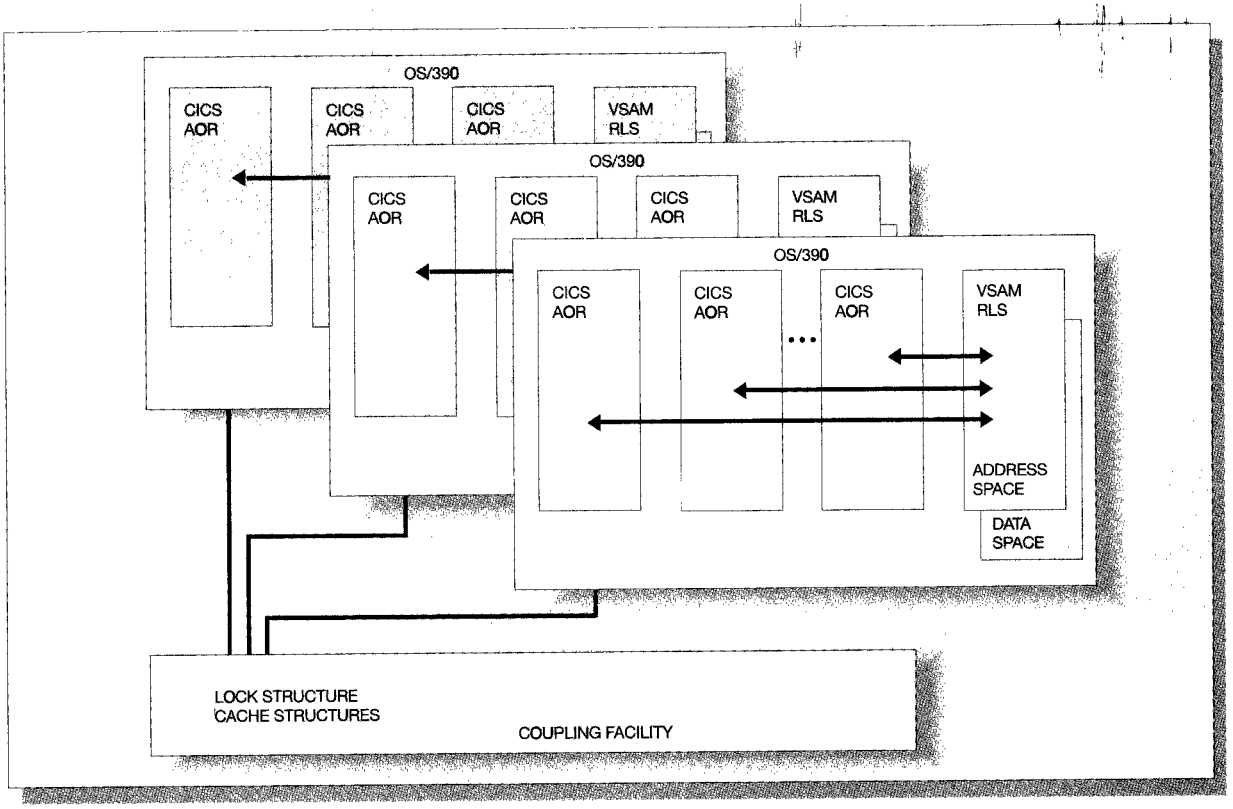
VSAM record-level sharing

A large amount of data exists as VSAM files accessed by CICS applications. This large volume of data and large application base generated the motivation for VSAM record-level sharing (RLS). The objective is to provide the full value of the Parallel Sysplex to these applications.

The heart of the S/390 Parallel Sysplex is its data-sharing technology based on the coupling facility. The transaction execution model assumes parallel (concurrent) execution of multiple transactions. The transaction isolation provided by the transactional recovery functions of the model assumes and enables data sharing. Three data managers that execute on OS/390* (formerly called MVS, or Multiple Virtual Storage), and provide transactional recovery for their data are: Information Management System Database (IMS-DB), DATABASE 2* (DB2*), and CICS VSAM. IMS-DB and DB2 have been extended to provide Parallel Sysplex data sharing by exploitation of the coupling facility. In the case of CICS VSAM, Parallel Sysplex data sharing is provided through changes to both CICS and VSAM.

RLS itself does not provide transactional recovery. When used through CICS, applications have the transactional recovery functions for the VSAM files pro-

Figure 3 RLS multisystem server



vided by CICS and the Parallel Sysplex data-sharing function provided by RLS. CICS provides logging and recovery. RLS provides locking and sysplex-wide parallel shared-data access.

VSAM RLS server. Previously, VSAM requests were executed within the caller's address space. The first step in the implementation of RLS was to build a continually running server address space. RLS requests submitted from any address space on the OS/390 image are handled by this server using a shared buffer pool resident within a data space. The RLS server provides the cross-memory access from the caller's address space to the server's address space. Figure 3 shows the OS/390 VSAM RLS environment. Multiple CICS AORs (application-owning regions) executing on each OS/390 image submit RLS requests to the RLS server instance resident on their OS/390 image. The multiple RLS instances use coupling facility structures to maintain integrity and consistency of the shared data.

In order to meet the availability requirements of a shared server, functional recovery was placed in all elements of RLS processing. Cancellation or failure of any thread under which an RLS request is executing results in the functional recovery performing cleanup to avoid interference with the execution of other RLS requests under other threads, tasks, and address spaces.

One instance of the RLS server is resident in each OS/390 within the sysplex. The multiple instances of the RLS server are aware of the existence of one another through the use of a sharing control data set, OS/390 cross-system coupling facility (XCF) messaging, and access to coupling facility lock and cache structures.

Before describing RLS serialization, let us briefly review non-RLS serialization. The term non-RLS means the application is not using RLS access mode. Instead, it is using either NSR (nonshared resources) or LSR (local shared resources) access mode. These are the

access modes that existed prior to RLS, and they continue to exist. There are now three choices: NSR, LSR, and RLS. Access mode is not an attribute of a VSAM file. It is an option specified at OPEN time.

Non-RLS serialization. Non-RLS VSAM provides serialization at the control interval level. The function is provided by the buffer manager. The buffer manager permits only one valid copy of a control interval to reside within the buffer pool. By obtaining exclusive use of a buffer, the owner has exclusive use of the control interval resident within the buffer. Of course, the scope of this serialization is the single buffer pool.

Occasionally it is necessary for VSAM to split a data control interval of a KSDS.

The owner of serialization on a buffer is a request parameter list (RPL) that an application set up to specify record access parameters for the VSAM request. The RPL represents a specific record access by an application. Some record accesses require exclusive serialization on the data buffer. If a second RPL requests exclusive serialization on a buffer that is held exclusively by another RPL, the request of the second RPL is rejected by VSAM.

Occasionally it is necessary for VSAM to split a data control interval of a KSDS. The split is required when data must be added to the CI and there is not sufficient free space within the CI for the new data. The split process moves some records from the old CI to a new CI. Even less frequently, a data control area must be split, which moves control intervals of data from the old control area to a new control area.

The control interval and control area split rely on the control interval level serialization of the buffer manager. The RPL that is adding the data already holds exclusive control of the CI. This control inhibits any other request (RPL) from modifying the CI while the split is in progress. Any such request fails with an error status being returned to the caller.

A control area split is triggered by a control interval split when a free CI is not available within the current control area. When non-RLS begins a control area split, the buffer manager attempts to obtain exclusive serialization of all control intervals within the control area. If any of the control intervals are exclusively controlled by another RPL, the request that needs the control area split is rejected. Once a control area split obtains exclusive control of all control intervals in the control area, no other RPLs can obtain exclusive control of any of these control intervals until the control area split completes.

In addition to serialization of the data control intervals, VSAM must serialize changes to the index control intervals of a KSDS. Non-RLS uses an exclusive latch to perform this serialization. This latch ensures that only one RPL at a time is modifying the index of a KSDS.

The split of a data control interval presents an additional problem. During the split process, there is a place where the records that were moved to the new data control interval also remain in the old data control interval and the index is updated to point to both control intervals. At this place, the key value in the index entry that points to the old data control interval does not agree with the highest key in that data control interval. If the job or task that is doing the split is cancelled or the OS/390 image fails at this time, there is an inconsistency between the data and the index of the KSDS. The KSDS architecture uses the CIDFBUSY flag to address this problem.

The data control interval processing sets the flag CIDFBUSY = ON in the old data control interval to indicate a split is in progress. This flag is set on before records are moved and the index is updated and remains on until the record movement and index update is complete. It is reset by the write command that eliminates the moved records from the old data control interval.

Now, the final part of the story. When a VSAM request accesses a data control interval and sees CIDFBUSY = ON, the request obtains the exclusive latch that serializes control interval splits. This causes the request to wait for completion of any split that may be in progress. After obtaining the latch, the request tests to see whether its buffer is still valid. If the buffer is not valid, it is released and the request is reprocessed to access the requested record. Finding the buffer still valid after obtaining the latch, VSAM compares the key of each record within the

data control interval with the key value in the index entry that points to the data control interval. When the key of a data record is greater than the key value in the index entry, the data record is removed from the data control interval. This cleanup process eliminates the data records that were moved to the new data control interval by the interrupted incomplete split.

RLS serialization

RLS uses the following serialization mechanisms:

1. Locking—Locking of individual records within VSAM data sets is a central element of the overall design. RLS also uses other higher-level locks for serializing operations such as control interval and control area split.
2. Buffer coherency—RLS uses an optimistic serialization protocol to achieve buffer coherency. The primitives used in this protocol are provided by the buffer validity test, buffer invalidate, and coupling facility conditional write functions of the Parallel Sysplex coupling technology.
3. DASD write serialization—RLS uses the castout lock functions of the coupling facility to achieve this serialization.
4. Control interval/area split serialization and CIDFBUSY flag—RLS uses an exclusive lock to serialize CI and CA processing for a KSDS. RLS uses the CIDFBUSY flag private serialization mechanism of the VSAM KSDS architecture to serialize requests to access data CIs while they are being split or moved by a control area split.

Recoverable files and nonrecoverable files. RLS provides data-sharing support for both recoverable and nonrecoverable files. The term recoverable file means that transactional recovery is provided for changes to the file. When a transaction requests rollback or the transaction abnormally terminates, all changes made by the transaction to recoverable files are backed out. The backout function for VSAM recoverable files is provided by CICS. Changes to nonrecoverable files are not backed out. When a change is made to a nonrecoverable file, the change is visible to other CICS and non-CICS applications as soon as VSAM writes the modified CI and releases the record lock.

RLS does not permit a non-CICS application to OPEN for output a recoverable file in RLS access mode. A non-CICS application may OPEN for input a recoverable file and may OPEN for input or output a nonrecoverable file.

RLS provides the data-sharing serialization mechanisms for both recoverable and nonrecoverable files.

Locking. The granularity of locking by RLS is a record within a VSAM data set. This minimizes lock contention across the multiple CICS transactions and other applications that are using RLS.

RLS uses two lock states. The shared state locks a record for read, and the exclusive state locks a record for write. A coupling facility lock structure is used to provide multisystem sysplex scope for the locks. Deadlock detection and resolution is provided for the locks. A time-out function is also provided whereby a VSAM request can declare a maximum amount of time that it is willing to wait for a lock.

RLS provides three read integrity options:

1. NRI (no read integrity)—This option accesses the record without obtaining a shared lock on the record. For a recoverable file, the reader may see an uncommitted change made by a CICS transaction. This form of read is sometimes referred to as a "dirty read." NRI does not incur the overhead of locking.
2. CR (consistent read)—This option obtains a share lock on the record. The CR request must wait if an exclusive lock currently exists for the record. A copy of the record is returned to the caller, and the share lock is released. Obtaining the share lock ensures that the record is not being updated or erased by another data-sharing transaction or application.
3. CRE (consistent read explicit)—This option is a form of repeatable read that is only provided for CICS applications. Like CR, CRE obtains a share lock at the time of the read. The share lock remains held until the CICS program completes. The record is inhibited from being updated or erased by other application instances until the instance that issued the CRE request completes. Although CRE inhibits update or erase of the accessed record, it does not inhibit the insertion of new records that satisfy the search criteria used to locate the accessed record.

Consistent read and repeatable read are new VSAM capabilities provided by RLS.

Lock hold duration for exclusive record locks. RLS obtains an exclusive lock on a record when the record

is being inserted, updated, or erased. When the change is made by a CICS transaction and the file being changed is recoverable, the exclusive lock must be held until CICS declares the lock is no longer needed. Normally, release occurs when the transaction reaches sync point. When CICS fails, or the RLS server fails, or OS/390 fails, the locks must remain held until the failed components are restarted and CICS has completed its transactional recovery processing for its transactions.

The requirement that RLS keep locks to protect uncommitted changes meant that VSAM needed to understand the recoverable attribute of a file. RLS introduces new VSAM data set attributes that declare a data set to be either recoverable or nonrecoverable. If recoverable, CICS provides transactional recovery for the data set. If nonrecoverable, transactional recovery is not provided.

For a nonrecoverable data set, RLS releases an exclusive record lock when the buffer containing the corresponding modified CI has been written. Since there is no transactional recovery, the change will not be backed out, and thus there is no need to continue to hold the lock. As mentioned earlier, RLS keeps exclusive record locks for recoverable data sets until CICS explicitly declares that they can be released.

Buffer coherency. RLS uses the buffer registration and invalidation functions of the coupling facility cache as the means to maintain buffer coherency across the local buffer pools of the individual RLS instances.

RLS uses the conditional write function of the coupling facility cache to implement an optimistic serialization protocol for changing data control intervals. The overall process is referred to as "record merge redo." We now describe the process.

The RLS locking granularity of records permits multiple concurrently executing transactions or applications to change different records that reside within the same data control interval. RLS assigns a separate buffer containing a copy of the data control interval to each of the sharers. The coupling facility local cache invalidate and conditional write functions are used to detect concurrent write activity at the control interval level. If the first write is successful, it invalidates the buffers assigned to the other sharers. This causes their subsequent attempts to write to their buffers to fail. When RLS detects this failure, it internally reaccesses the data set and reapplies the change. This process merges the change with the earlier changes and is the record merge redo. A record merge redo example is shown in Figure 4.

DASD write serialization. RLS utilizes the coupling facility cache as a read/write cache. This use means that data control intervals and index control intervals are placed in coupling facility caches by VSAM. The coupling facility cache serves as a new level in the storage hierarchy for these data.

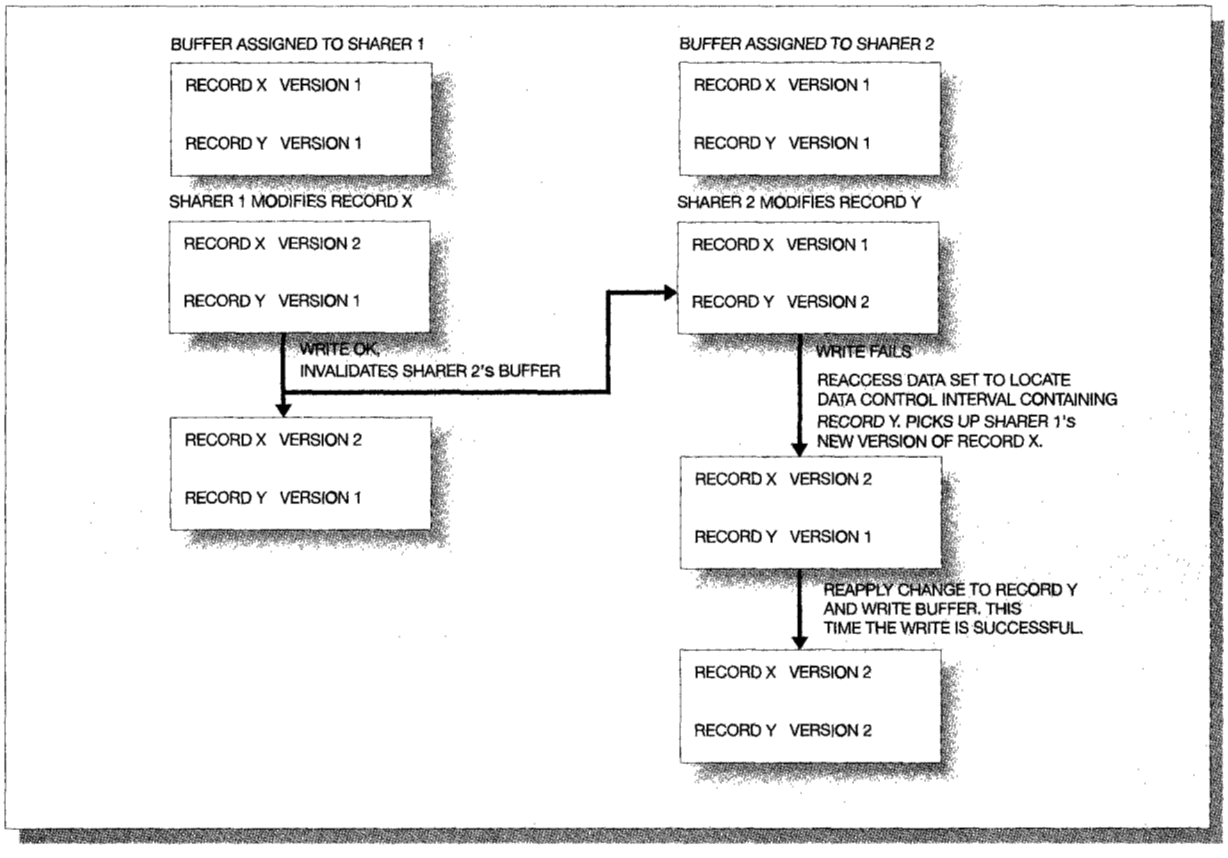
To avoid loss of data in the event of coupling facility failures, RLS uses store-through protocols when writing data to a coupling facility cache. This means that when RLS writes modified data to a coupling facility cache, it immediately writes the same data to DASD. RLS inhibits access to the modified data until they have been successfully written to DASD.

First, serialization is required to ensure that from the time a modified control interval is written to a coupling facility cache until the change has been written to DASD a second writer does not get in and make a second change that would be lost (overlaid) by the first writer's DASD write. The coupling facility provides a special "castout lock" function at the cache entry level. RLS requests the castout lock to be set on the cache entry before writing the control interval to DASD. If the castout lock is already set on the entry, the request fails. Otherwise, the cache write atomically writes the data to the coupling facility cache and sets a castout lock on the cache entry that contains the data. Until RLS issues a coupling facility request to unlock the castout lock, an attempt by another process (concurrent writer) to write to the castout locked entry fails.

In the event that an instance of RLS fails while having castout locks set on entries in coupling facility caches, the abnormal disconnect from the caches results in the coupling facility deleting the cache entries that are castout locked by the failed instance. This deletion ensures that data that may have been written to the coupling facility cache but not written to DASD are not used. No loss of any data results. Since the castout lock was not released, RLS does not complete the write.

In addition to blocking new writes of the data until both the coupling facility write and the DASD write are complete, RLS blocks reads of the new version of the control interval. The read blocking is accomplished as follows. The coupling facility returns the setting of the castout lock as feedback on a read of the corresponding entry from the cache. When RLS

Figure 4 RLS record merge redo—example of RLS handling of the case where two sharers concurrently modify two different records that reside within the same data control interval



sees the entry is castout locked, it does not use the data. Instead, it sets a timer and waits for the time interval to expire. In this way the write operation has time to complete and the castout lock has time to be reset. The coupling facility read is reissued.

Control interval/area split serialization and CIDFBUSY flag. Since control interval/area splits are rare, RLS uses a data set level lock for serialization. Only one split at a time is permitted for a specific KSDS, and any interference between multiple splits is avoided.

A design challenge in the development of RLS was how to serialize record access across control interval/area splits. The basic problems are:

- One or more records that need to be moved by the split may be locked for change by other transactions or applications.
- While the split is in progress, concurrent access to a record that is being moved by the split must not receive a false record-not-found status.
- A new record that is added to the control interval/area that is being split while the split is in progress must be placed properly within the data set. The result must be the same as if the split was completed before the new record was inserted.

The first RLS design decision in addressing these problems was to not use control interval level locking in either the record access processing or the split processing. This decision avoided control interval level contention and possible deadlock across concurrent record accesses and split processing.

The lock name for a record in a KSDS is derived from the key value of the record. Using a key-based name instead of a record position (relative byte address) name for the lock allowed split processing to move

a record without affecting a lock that might be held on the record. The split processing does not obtain any record locks. It simply moves records. A record that is being moved by the split process may be locked by another process that is accessing the record. The record lock may be held across the split or may be obtained while the split is in progress.

Designing the RLS split processing so that it does not obtain record locks avoided any record locking conflicts across record accessing and the split. This enabled concurrent access to records during split processing.

The remaining problems are how to avoid false record-not-found conditions when attempting to access records that are being moved by a split, and how to properly insert new records that fall within the range of the split while the split is in progress.

RLS use of CIDFBUSY. Like non-RLS VSAM, RLS uses CIDFBUSY to indicate that a data control interval split is in progress. The control interval split latch used by non-RLS is replaced by RLS with a control interval split lock. This exclusive lock serializes RLS control interval splits for a KSDS. Just as with non-RLS, an RLS split may not complete due to a cancellation or a failure. RLS contains logic similar to non-RLS to eliminate duplicate records left in the old data control interval by an interrupted incomplete split.

Since RLS does not perform control interval level locking, it has the problem of serializing access to data control intervals that are being moved by a control area split. RLS uses the CIDFBUSY flag as the solution to this requirement. The control area split processing sets CIDFBUSY = ON in each data control interval that is to be moved to the new control area. The processing holds the control interval split lock prior to setting these flags. A concurrent read or write request that accesses one of these control intervals sees CIDFBUSY = ON and requests the control interval split lock. This causes the request to wait until the control interval split lock is available. Concurrent access to the data control intervals that are being split or moved by a split is serialized, and the two problems listed above are solved.

Broadcast of data set space allocation or usage

Each instance of the RLS server maintains information in virtual storage about control blocks data set extent and current end-of-used-space (high-used space). When one instance of the server modifies this information, the change must be broadcast to other instances of the server that have a copy of the information. RLS uses XES locks as the method of performing the broadcast. Each RLS instance holds a special lock on each linear space of each data set that it has in use. When the space allocated or used information for a liner space changes, locking protocols send the changed information to each RLS instance that holds a lock on the linear space.

Conclusion

RLS uses the System/390 coupling technology to provide record-level data sharing. Although RLS provides locking and buffer coherency to achieve data integrity for the shared data, it does not provide the application level or transaction level isolation and change backout functions of transactional recovery. The transactional recovery functions are provided by CICS, which has been extended to support RLS. Together, CICS and VSAM RLS provide the benefits of the System/390 Parallel Sysplex to CICS VSAM applications.

*Trademark or registered trademark of International Business Machines Corporation.

General references

DFSMS/MVS Version 1 Release 3 DFSMSdfp Storage Administration Reference, SC26-4920, IBM Corporation; available through IBM branch offices.

DFSMS/MVS Version 1 Release 3 General Information, GC26-4900, IBM Corporation; available through IBM branch offices.

DFSMS/MVS Version 1 Release 3 Macro Instructions for Data Sets, SC26-4913, IBM Corporation; available through IBM branch offices.

DFSMS/MVS Version 1 Release 3 Using Data Sets, SC26-4922, IBM Corporation; available through IBM branch offices.

Accepted for publication January 10, 1997.

Jimmy P. Strickland *IBM Storage Systems Division, 5600 Cottle Road, San Jose, California 95193-0001 (electronic mail: jimmy_strickland@vnet.ibm.com).* Mr. Strickland is an IBM Senior Technical Staff Member. He is an architect for storage systems software products and was a member of the technical team that defined the architecture of the S/390 Parallel Sysplex.

Reprint Order No. G321-5648.