IBM

Washington Systems Center

Technical Bulletin

The JES2 Checkpoint Mechanism

Written by: P. A. Eden
DSD Poughkeepsie

Published by: J. M. Hutchinson
Washington Systems Center

GG22-9220-00
December 1980

Washington Systems Center
Gaithersburg, Maryland
Technical Bulletin

The JES2 Checkpoint Mechanism

P.A. Eden, DSD, Poughkeepsie
Published by J.M. Hutchinson

GG22-9220-00
December 1980

This Technical Bulletin is being made available to IBM and customer personnel. It has not been subject to any formal review and may not be a total solution. The exact organization and implementation of the functions described will vary from installation to installation and must be individually evaluated for applicability.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to: IBM Washington System Center, 18100 Frederick Pike, Gaithersburg, MD 20760.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

ABSTRACT

This document is a result of a presentation made by Phyllis Eden at a GUIDE user group meeting in November, 1979. It is being made available in this Washington Systems Center Bulletin for wider distribution to IBM representatives and customers.

THIS PAGE INTENTIONALLY LEFT BLANK

CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ILLUSTRATIONS

THIS PAGE INTENTIONALLY LEFT BLANK

THE FUNCTION OF THE JES2 CHECKPOINT

Originally, in HASP and in early versions of JES2, the sole purpose of the checkpoint was to maintain a copy of the work queues on DASD so that the system could be restarted. Periodically, the storage copy of the job and output queues were written to the DASD, hence the name 'checkpoint'. When the system was restarted after a failure or normal shutdown, HASP could read the checkpoint from the spool and continue processing as if it had never been interrupted.

When JES2 Multi-Access Spool was developed to allow loose coupling of processors, the checkpoint found a new function as the communication pathway between the processors. A Multi-Access Spool complex consists of from two to seven processors (members), each running asynchronously but cooperating. In a JES2 Multi-Access Spool complex all the processors share access to the same set of job and SYSOUT queues. A job can be read in by any processor, can execute on any processor, its SYSOUT can be printed or punched by each of the processors, and operators can control jobs anywhere in the complex. Each processor in a Multi-Access Spool complex maintains an in-storage copy of these job and SYSOUT queues. All processors in JES2 Multi-Access Spool are therefore equal in control and in processing responsibility; in such 'peer-coupled' systems there is no master/slave relationship.

As the main mechanism for communication between processors of a Multi-Access Spool complex, the checkpoint data set contains much of the information which JES2 needs in order to control its functions.

As previously alluded to, the checkpoint contains all of the job queues and SYSOUT queues from which work is selected. Aside from these the checkpoint contains several other important structures which control its own operation and related aspects of JES2. For example, the checkpoint contains a record of system values describing the overall configuration of the Multi-Access Spool environment and specific characteristics and information describing the current status of each member system.

The checkpoint data set also contains the information that the processors in the Multi-Access Spool complex use to 'share' the spool space. This includes extent information about the spool volumes which are currently mounted and the bit map from which spool space is allocated. In fact, structures in the job and SYSOUT queues contain pointers through which all data on the spool volumes is actually located.

The checkpoint also plays a role in both RJE and NJE operations. It is used as an 'anchor' for RJE/NJE message queues and also contains a bit map which indicates whether and to what systems remotes are attached.

JES2 Checkpoint DASD

The checkpoint data set resides on a shared DASD which is accessible to all the processors in a Multi-Access Spool complex. By reading and writing information from/to the checkpoint data set, a processor maintains an up-to-date copy of the checkpoint information in its storage. When a processor changes information in the queues, it must reflect the change to the other processors via the checkpoint data set. 'Ownership' of the authority to change the checkpoint information is controlled using the RESERVE/RELEASE feature of shared DASD. RESERVE/RELEASE also ensures the consistency of the checkpoint data by preventing concurrent updates.

JES2 Checkpoint Data Set and Control

Since the JES2 checkpoint is a very frequently accessed data set, the performance of its I/O is a key consideration. For example, JES2 uses EXCPVR for the checkpoint to reduce the overhead associated with each I/O by translating its own channel programs and by performing its own page fixing. The actual format of the data set (after the application of the fix for APAR OZ27300) is non standard. That is, the data set is composed of both keyed and nonkeyed records. The records containing the job and output queues are nonkeyed; however, the first track of the data set now contains a special keyed record which is used to control access to the data set. The checkpoint data set is generally relatively small. Its size is dependent on some initialization parameter values which determine the size of the job queue and the output queue. However the maximum size of the data set, given the sizes of job and output queue elements today, is only one track, plus up to 102 4K records.

One might think of the JES2 checkpoint as a kind of 'shared virtual storage'. Each processor contains a copy of the same information, and this information, which is divided into 4K pages, is read from and written to a DASD data set based on requirements for references and changes. The JES2 checkpoint data set (SYS1.HASPCKPT) can be thought of as the JES2 special paging data set. Software facilities similar to the hardware change bits have been implemented to maintain up-to-date copies of the shared information on DASD and in all processors. The shared virtual storage analogy is actually a very good one; in fact, most of JES2 (everything except the checkpoint processor) deals with the mechanism in exactly this way. Other processors within the system are never aware of the I/O operations associated with the checkpoint.

The similarity of checkpoint processing to the MVS paging mechanism is apparent. The JES2 checkpoint processor contains mechanisms that delay processes referencing checkpoint data until an up-to-date copy of the information is available in that system's storage, analogous to the page-fault and page-in operations. JES2 also provides a corresponding mechanism analogous to page-out. When a change is made to a 4K checkpoint 'page', it can be flagged as changed so that it will automatically be written out by the checkpoint processor. These flags, called control bytes are actually part of the data set and also serve as an indication to other processors of which pages must be read in.

Unlike the virtual storage management of the operating system, JES2 has no hardware facilities available to 'trap' references or record changes to the checkpoint data.

Before any JES2 routine updates an in-storage copy of a checkpoint page, it must be sure that its processor has exclusive control of the checkpoint and that that page is current. To do this, a JES2 routine may issue a $QSUSE macro instruction that calls a service routine which ensures that all JES2 checkpoint data is locked onto that system, preventing other processors from updating it. Since one of the other systems may currently own the checkpoint, this may result in a $WAIT, until the owning system releases the checkpoint. Once the $QSUSE macro is complete, the routine is free to update any checkpoint information until it issues a JES2 $WAIT (explicitly or implicitly, e.g., $GETBUF WAIT=YES), at which time another $QSUSE macro is necessary to ensure ownership before updating.

Once the JES2 routine has updated some checkpoint information, it must then inform the checkpoint processor and other systems that a change has been made. To record changes to checkpoint pages, JES2 uses flag bits maintained in control bytes. For each 4K record in the checkpoint data set, there is a corresponding control byte. The $QCKPT and $#CKPT macros are used when a change is made to a page. These macros set the corresponding control byte. Since the control bytes are also recorded on DASD and

available to each processor, they may also be used during checkpoint read operations to determine which pages have been changed and must be read to maintain a current copy in storage.

The JES2 Checkpoint Cycle

An overview of JES2 checkpoint processing shows that there are four basic stages during a checkpoint cycle. The first phase of the checkpoint cycle begins with a RESERVE operation to ensure exclusive control of the checkpoint data set. Next, using information from the control bytes, any records changed by other processors are read.

In the second phase of checkpoint processing, a system is said to 'own' or 'hold' the checkpoint. It can perform processing on and make updates to the checkpoint data. During this phase, the processor can cause intermediate versions of the checkpoint information to be written to the DASD without losing its ownership.

At the completion of the hold interval, the final write of the data set is made and a RELEASE operation is performed which enables the next processor to start its checkpoint cycle.

The final stage is a dormant period during which a processor makes no attempt to access the data set and thereby allows other processors time to complete their active phases.

The length of time in each stage of the checkpoint cycle is determined by the installation, using three initialization parameters:

> &MINHOLD specifies in hundredths of a second the minimum length of time a member of a Multi-Access Spool complex must maintain control of the shared queues after gaining control of them. This parameter is related to the amount of processing time a given system may spend examining and updating the checkpoint information. As such, it is somewhat related to the processor's relative speed in the Multi-Access Spool complex.
>
> &MINDORM specifies in hundredths of a second the minimum time a member of a Multi-Access Spool complex must wait after releasing control of the shared queues before again attempting to gain control of them. This parameter provides a means of preventing one member from monopolizing control of the shared queues, and allows adequate time for the other processors to complete their &MINHOLD intervals. Since the object is to avoid contention for the checkpoint data set, the &MINDORMs of each processor in the complex should be large enough to allow for the sums of the other processors' &MINHOLD values.
>
> Finally, &MAXDORM specifies in hundredths of a second the maximum time a member of a Multi-Access Spool complex may wait before it is required to read the checkpoint data set. When JES2 is idle, this parameter ensures that it periodically looks at the shared queues for eligible work that another member of the complex may have placed there and maintains a reasonably up-to-date copy of the queues in its storage.

There are two basic modes of operating a Multi-Access Spool complex. The most prevalent is in a controlled environment where each processor 'gets its turn' owning the checkpoint data set. The other, contention mode, is not recommended. In contention mode, it is possible to allow the processor to compete for the checkpoint at all times by specifying &MINDORM and &MINHOLD equal to 0.

Since the JES2 checkpoint data set can be accessed by several processors, the RESERVE/RELEASE feature of shared DASD is used to control access to it. When a particular JES2 system wishes to access the checkpoint, a RESERVE is issued. This allows this processor to update the data set until it issues a RELEASE. If another processor attempts to access the device on which the checkpoint data set resides while the first processor is still holding the RESERVE, the second processor will be returned a busy condition to its I/O operation. When the processor holding the RESERVE finally issues the RELEASE for the DASD, an interrupt (device end) is signalled to all processors which experienced the busy condition. At this point, the DASD is unlocked, and the other systems can again try to reserve it.

JES2 Checkpoint I/O Operations

As shown earlier, the checkpoint cycle begins with a read operation to update the in-storage queues. Each read operation is divided into two separate I/Os. The first issues a RESERVE for the checkpoint device and reads control information including the control bytes from the first track. The control bytes are then used to build a channel program to read in all of the records which were changed by other processors.

Because performance is a primary consideration with respect to checkpoint I/O, JES2 uses EXCPVR and manages its own real storage. The processing that JES2 uses to read the checkpoint is depicted in this slide. In addition to the storage actually used to contain the checkpoint data, JES2 maintains a checkpoint I/O buffer. During normal checkpoint I/O, actual CCWs transfer data to this buffer. First (step 1), using the control bytes, JES2 'fixes' (PGFIX with the RLSE option) the real frames associated with pages in the buffer that will be used. After a channel program has been built and executed (step 2), JES2 moves (step 4) each of the read pages to the appropriate area of the actual checkpoint storage. However since it will be replacing data in these pages with the MVCL, it releases the old data (PGRLSE) in the associated frames or ASM storage slots first (step 3). Finally it can release the frames used as an I/O buffer. This operation (steps 3, 4, and 5) is performed in a loop, a page at a time, so that the real frame requirement for the I/O never exceeds by more than one the total number of pages read.

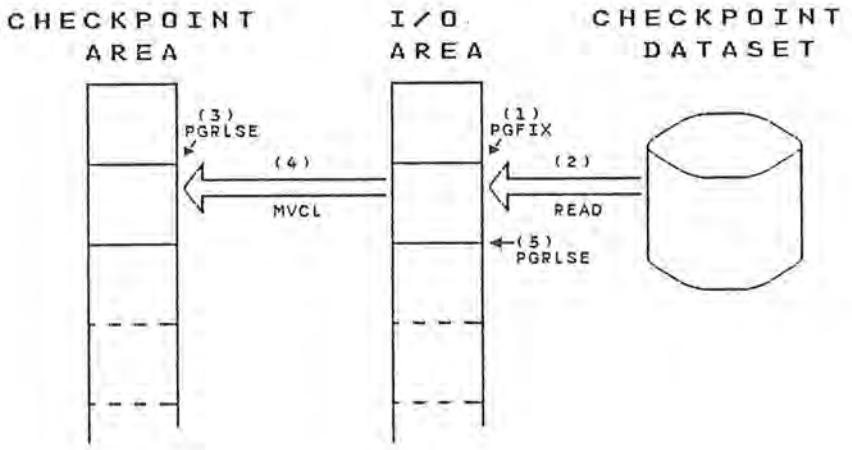

Figure 1. JES2 Checkpoint Read Operation

Once a processor has control of the checkpoint data set, it is free to update it until its checkpoint interval expires. Each of these updates involves writing the master record, which includes the control bytes which indicate the changed records as well as the changed records themselves. When the checkpoint interval ends, the processor will make its final update to the data set and then release it.

All the write operations performed by the checkpoint processor use the same general algorithm. The operations are in fact the reverse of the ones used by read. First, as in read, each page of the I/O buffer that will be used in the EXCPVR is fixed (PGFIX with the RLSE option). For each page in the checkpoint data set that was changed, the storage in the actual checkpoint area is moved to the fixed page in the I/O buffer (step 2), and real addresses in the channel program are adjusted. Next the EXCPVR is issued (step 3), and the changed pages are written to DASD. Finally the pages in the I/O buffer that were fixed are released (step 4).

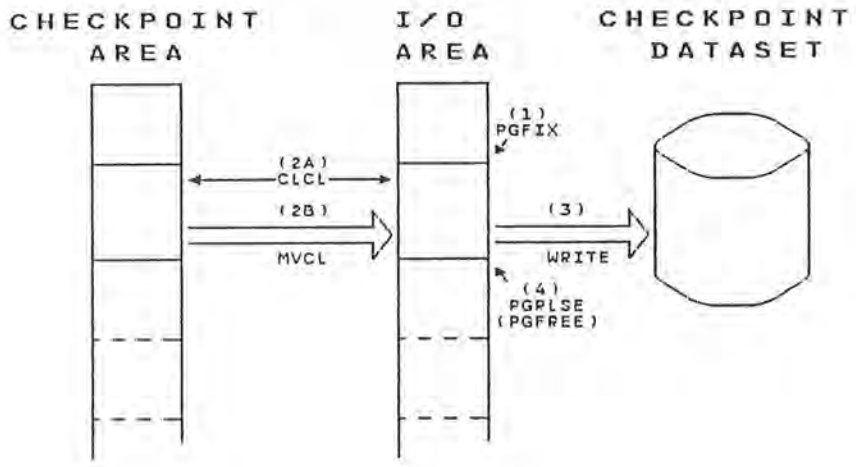

Figure 2. JES2 Checkpoint Write Operation

In some cases in both the read and write cycles, these sequences are slightly altered to allow special validity checking associated with the &DEBUG Initialization option. In this case the I/O area is never actually released, but simply page freed after read. During the write cycle, pages whose control bytes do not indicate change are compared with the saved copy in the I/O area to ensure that an error has not occurred (step 2a). For example, the two copies of the page might not be identical if an update was made without a $QCKPT or $#CKPT.

HISTORY OF THE CHECKPOINT FUNCTION

The checkpoint function has a long history and recently has undergone several major redesigns.

The checkpoint function has existed as far back as HASP (late 1960s). In HASP, and in early versions of JES2, the first few tracks of the spool were formatted into special checkpoint records. This information was used primarily for warm starting.

With MVS and JES2 Release 3, Multi-Access Spool was introduced. In Multi-Access Spool, the checkpoint serves as the communication mechanism between processors, and some hardware control over the updating of information was needed. Therefore the checkpoint information was moved to a separate MVS data set.

As installations grew, the size of their checkpoint data sets also grew and some Multi-Access Spool installations were beginning to experience performance problems related to the amount of time required to perform checkpoint I/O operations. Although JES2 would only rewrite changed checkpoint records, it had to read the entire data set because there was no way of determining which records had been changed by other processors. As a result, JES2 could become essentially serialized until checkpoint I/O

completed. This led to RJE timeouts and slow responses to operator commands. Also, much of the I/O for the checkpoint data set was unnecessary, because many of the pages in the job queue and job output table were unchanged. Thus the records being read overlayed identical data already in storage.

This performance problem was addressed with the fix to APAR OZ20010. With this fix, control bytes which reside on the checkpoint data set itself are used to identify which records were changed so that only those selected records need to be read in. The physical format of the data set was changed to consist of fixed length 4K blocks. In addition to the selective read capability, EXCPVR was chosen as the access method in order to improve performance even further. Also, JES2 began using the services of MVS to manage the real storage associated with checkpoint I/O buffers.

After about a year, reports of several major unrecoverable checkpoint failures were received. Although these failures were induced by severe hardware conditions, it was apparent that JES2 did not provide adequate error recovery for its checkpoint data set. Clearly the loss of a checkpoint data set could be a major problem in large Multi-Access Spool installations and even somewhat uncomfortable for a single processor. Since the JES2 checkpoint represents an important system resource, it became obvious that significant (in fact, extraordinary) measures should be taken to protect it.

The fix for APAR OZ27300 provides a variety of both simple and complex schemes for error detection, retry, recovery, and avoidance.

In addition to taking advantage of normal MVS error recovery, JES2 retries even those errors considered permanent by ERPs. Special error detection mechanisms, including unusual channel programming and a philosophy of minimal dependence on hardware error reporting, were implemented. An auxiliary locking mechanism to supplement the shared DASD RESERVE/RELEASE facility was invented. Most importantly a new secondary copy of the checkpoint data set is now optionally maintained as a backup in case of media damage or other failures. Finally, additional processing was included in initialization to allow JES2 to repair minor structural damage to the checkpoint and continue with minimal loss of job and SYSOUT data.

JES2 CHECKPOINT INTEGRITY

First let us examine the problem and its ramifications. The JES2 checkpoint data set can be rendered unusable for a wide variety of reasons, ranging from hardware and software errors to operational mishaps. At one extreme, the data set can be affected by the loss of the DASD volume, due to a 'head crash' or other physical damage. On the other hand, more recently it was discovered that certain operational procedures, which are used generally to avoid a system crash, may also at times be responsible for allowing invalid data to be written to the checkpoint data set.

Customers with large Multi-Access Spool complexes are the ones which experience these types of failures most frequently. They are also the ones which are most seriously affected by a complete loss of their checkpoint data set. Regardless of size, loss of a checkpoint data set is a serious event in any installation today.

The amount of queued work in terms of input jobs and SYSOUT lost can be significant in both processor hours and printed lines of output. All of the jobs in the input queues and all of the SYSOUT data queued are gone.

Another serious consequence is the interruption in service. All the systems in the complex must be re-initialized, and the JES2 checkpoint data

set must be reformatted by a 'cold start'. In today's installations with large numbers of interactive users, far-reaching networks, and large numbers of RJE stations, the 'down-time' necessary to re-initialize all of the systems in the complex is a problem. However, even when the systems are all running, the installation has to determine what work was in the queues, re-enter the jobs which had not yet executed, and rerun the jobs which had output queued, if possible.

Checkpoint Integrity Philosophy

Because of the impact of this problem, the JES2 checkpoint integrity fix was designed with several goals in mind. First, if possible, any interruption to service should be avoided, that is, JES2 should not fail unnecessarily. Second, if it is necessary to take JES2 down, only a warm start should be required. In the best case this will allow JES2 to continue with no loss of jobs or of SYSOUT. However even if some work must be lost, the bulk of the job and output queues will remain intact. Third, when it is necessary to continue operation with a slightly damaged checkpoint data set, JES2 initialization should detect and repair errors in its structure rather than expose the system to the possibly snowballing effects of these errors. Fourth, in cases where both the primary and secondary checkpoint data sets are damaged beyond use, JES2 should allow the system programmer access to the checkpoint information recorded in a SYS1.DUMP data set, for possible partial recovery of the larger jobs in the system and for diagnosis of the problem. Finally, JES2 must provide sufficient error messages and descriptions to allow operations personnel and system programmers to determine the nature and severity of checkpoint problems and decide on appropriate actions.

The checkpoint integrity functions in JES2 have been implemented to provide protection from a variety of possible failures.

In addition to retrying simple transient errors, JES2 will now attempt to detect and recover from errors heretofore considered undetectable by the hardware.

The single most important mechanism for recovery from severe failures is the availability of a secondary copy of the checkpoint data set.

Also, a new hardware/software locking strategy for the checkpoint data set is used as a backup for the RESERVE/RELEASE feature. Unlike RESERVE/RELEASE, this mechanism is unlikely to fail in a way that could affect the integrity of the checkpoint.

Finally, it is now possible for either the issuing processor or other processors in the Multi-Access Spool complex to detect channel programs which are interrupted and not normally completed.

The Alternate Checkpoint Data Set

One of the most significant external changes to the JES2 system that was effected by the checkpoint integrity changes is the definition of a new and optional facility that allows an installation to maintain a secondary copy of the JES2 checkpoint data set. This data set, called the 'alternate' or 'duplex', is an exact image of the primary data set that resides on a separate DASD volume. The volume serial of the disk which contains this data set is defined via an initialization parameter (&CHKPT2). However, the requirement that the alternate data set not reside on the same volume as the primary is the only special consideration given

to it, other than its accessibility. It does not have to be on the same device type as the primary data set.

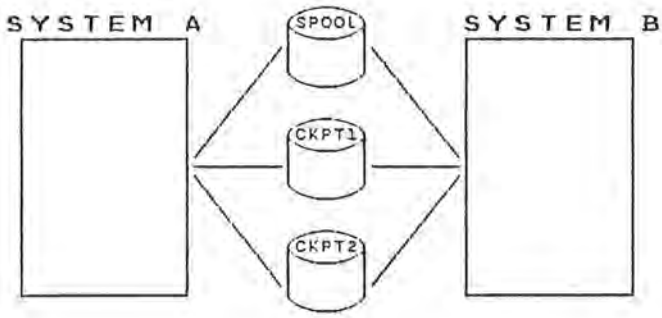

Figure 3. Alternate Checkpoint Data Set

If the primary data set is damaged, an initialization option can be specified to use the alternate data set to 'warm start' the system. If the primary DASD volume has been damaged beyond use, for example when a head crash has occurred, the alternate checkpoint can even be used as a replacement for the primary.

In situations where the physical DASD on which the primary data set resides has not been damaged, it might be possible to restart the system with both checkpoint data sets in the configuration and using the alternate as the source of the checkpoint data. The "ALTCKPT" initialization option allows the installation to warm start from the duplex checkpoint. In this case, the checkpoint information will be read from the alternate and used to refresh the primary. After initialization is complete, both the primary and alternate checkpoint data sets will be used in their original roles. Because the primary checkpoint data set is rewritten, the use of this initialization option is allowed only during a complex-wide warm start.

The alternate data set can be used in either a Multi-Access Spool complex or in non-loosely coupled environments. It is not necessary for each processor in the complex to have access to the duplex data set. Some installations may want to configure so that only some of the processors, perhaps only one, will perform the checkpoint duplexing function. However, it is recommended that no fewer than two duplex.

There are certain characteristics of the I/O for the alternate data set which are of interest. JES2 always checks to see that I/O to the primary data set has completed before initiating the channel program for the duplex. This ensures that at least one of the data sets will be valid should an incident, such as a power failure, occur during an update operation. As mentioned earlier, if the primary checkpoint is damaged, the alternate can always be used to restart. When this is done, the checkpoint data is read in from the duplex. This is the only place where the alternate checkpoint is read. During normal operation it is only written. Since writes to the alternate data set are always synchronized with I/O to the primary, the locking mechanism for the primary can also be used to safeguard the data in the duplex. This means that the RESERVE (and lock) which is used for the I/O to the primary checkpoint is used to guarantee that no other processor is updating the alternate data set. For this reason, it is important that no other data sets which might be shared between the sys-

tems be placed on the same volume as the alternate data set. An interlock can result if this is done. For example, system B reserves a load module library on volume CKPT2 (the alternate checkpoint volume) while system A owns the RESERVE (and lock) on the primary checkpoint volume CKPT1. System B then tries unsuccessfully to reserve the primary checkpoint volume CKPT1 while processing in a JES2 subsystem interface (HASPSSSM) service routine for the same task that issued the RESERVE for CKPT2. A system deadlock will occur. System A holds the primary RESERVE and will not release it until it has successfully written the alternate, which it is unable to do due to system B's outstanding RESERVE. System B, on the other hand, will not release its RESERVE on the secondary checkpoint volume until it exits the JES2 service routine, after acquiring the primary RESERVE.

Because the alternate is the main source of recovery data in the event of a failure, JES2 uses all of the extended error detection and recovery facilities which are available with this fix for both the primary and alternate checkpoint data sets.

Because the alternate checkpoint data set provides the only backup for the checkpoint data, there are some recommendations for its placement. The system will ensure that the primary and alternate data sets are on separate volumes. Second, if it is possible, these data sets should have separate DASD control units, separate power sources, and separate channels or I/O directors. A further step in ensuring the validity of the checkpoint data is to physically isolate the DASD units, so that environmental hazards, such as fire, flood, or physical impact, cannot damage both devices.

Checkpoint Data Set Format

In order to implement several of the schemes associated with extended I/O error recovery and the new data set lock, the format of the checkpoint data set has changed. There are two additional records on the first track.

The first track of the checkpoint data set and the channel programs associated with it have been designed to assure that any DASD device type can be used for either the primary or the alternate data set and to minimize the associated rotational delay. Slightly more than one revolution is required in order to execute the channel program which performs the initial read of the records on the first track when a checkpoint cycle begins.

As mentioned earlier, the checkpoint data set is now a non standard data set, containing both keyed and nonkeyed records.

The first track now contains three control records: an 8-byte check record, a lock record composed of an 8-byte key field and an 8-byte data field, and the master record. The job queues and output queues are segmented into 4K records which reside on the remainder of the data set. There is a possibility that some portion of the last 4K record in the job queue and in the job output table may not be used, since the sizes of checkpoint structures are rounded to fit into 4K boundaries. The format of the duplex checkpoint data set, if one exists, is identical to the primary.

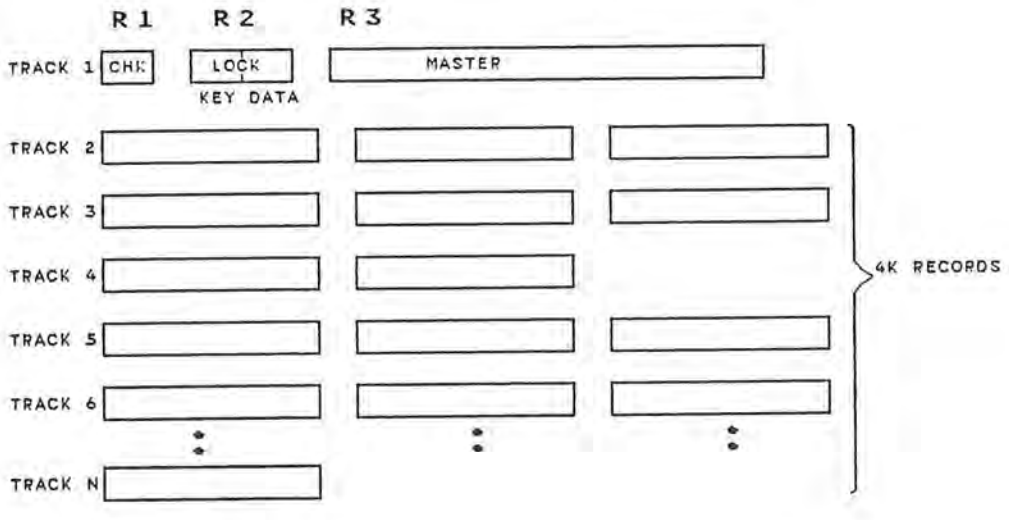

Figure 4. Checkpoint Data Set Format

The check record is an 8-byte record at the beginning of the first track of the checkpoint data set. It contains a check value used to help determine whether the remainder of the data in the checkpoint is valid. That is, it will be used in conjunction with a companion value in another record to indicate whether the previous update operation completed successfully.

The second record on the first track is the lock record, the only keyed record in the checkpoint data set. It consists of an 8-byte key portion and an 8-byte data portion which are identical. It is used as a software lock in addition to the normal hardware RESERVE/RELEASE mechanism.

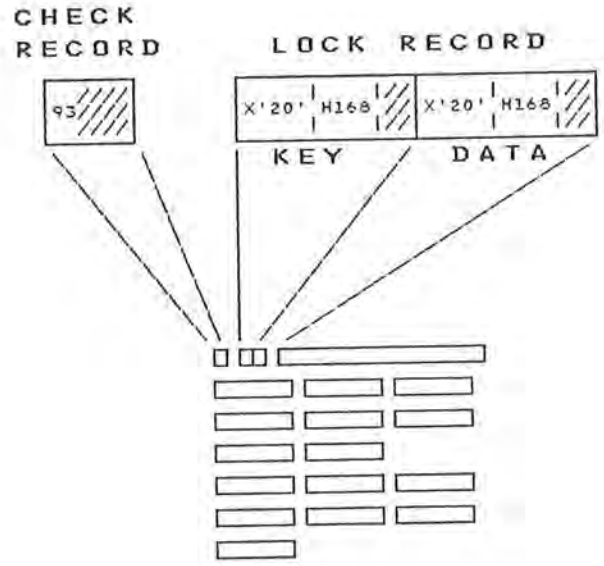

Figure 5. Track 1 - Check and Lock Records

The JES2 master record is the third record on the first track of the checkpoint data set. It contains information from selected variables of the JES2 HCT. Among these values are the initialization parameters which affect the configuration of the Multi-Access Spool complex ($SAVEBEG); the QSE data areas that represent the status of each processor in the complex; the checkpoint control bytes ($CTLB); a series of bytes that describe the extents of currently mounted spool volumes ($DACKPT); an RJE sign-on control table ($RMTSON); the RJE/NJE message queues ($MSPOOLQ); and the master Spool track allocation bit map ($TGMAP). Importantly, among the variables in the first part of the record ($SAVEBEG) is the copy of the check value which is compared to the value in the check record during read operations to determine whether the records in the checkpoint data set are all at the same update level. This will be elaborated on later.

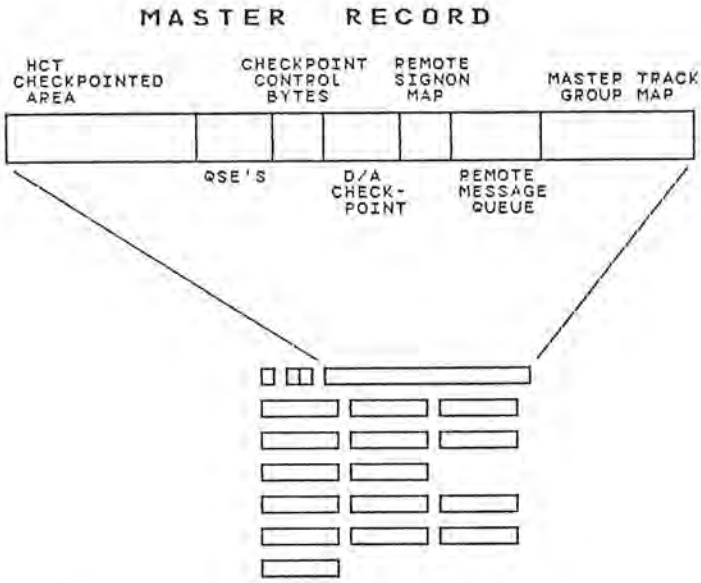

Figure 6. Track 1 - Master Record

Checkpoint Data Set Lock

The checkpoint data set lock is used as a backup for the RESERVE/RELEASE feature of shared DASD. RESERVE/RELEASE, by itself, is not an adequate mechanism to guarantee that simultaneous updates will not occur, because it has a tendency to open the lock, unintentionally, when failures occur. The checkpoint data set lock provided by JES2, on the other hand, tends to lock closed under these conditions. It will always ensure that the data in the checkpoint data set is good by prohibiting simultaneous updates under any circumstances. Because of this characteristic of locking closed when a failure occurs, this lock requires a manual operation to reset it.

When a processor gets control of the shared checkpoint, it will write a system-dependent value into the key and data portions of the lock record. When the lock is not held by any processor, a value of zero will be recorded. Processors in the Multi-Access Spool complex can determine whether the shared data set is available by using a SEARCH KEY EQUAL channel command with a zero data field. If the SEARCH KEY EQUAL operation is successful, the remainder of the channel program will set the key field to the appropriate value for the processor. This channel program is basically a 'compare-and-swap' operation on the lock record.

The operation to obtain the lock will be done as part of the initial read channel program which is executed as each processor's checkpoint interval begins. For the lock to function correctly in all circumstances, it is imperative that the initial operation that compares the lock value (SEARCH KEY EQUAL) not be separated from the operation of writing the lock. For this reason these two operations are a part of a single channel program. If an error occurs during this initial read channel program, MVS error recovery cannot be allowed to function, since it might attempt to restart the channel program at a failing CCW in the middle.

```
        CCW   SRCH+ID+EQ,LOCK,CC,5
        CCW   TIC,*-8,0,0
        CCW   SRCH+KEY+EQ,ZERO,CC,8
        CCW   TIC,FAIL,0,0
          .     .
          .     .
        CCW   SRCH+ID+EQ,LOCK,CC,5
        CCW   TIC,*-8,0,0
        CCW   WRITE+KEY+DATA,SYSID,CC,16
          .     .
          .     .
        CCW   NOP,0,0,0
FAIL    CCW   SRCH+ID+EQ,LOCK,CC,5
        CCW   TIC,*-8,0,0
        CCW   READ+KEY+DATA,AREA,CC,16
```

Figure 7. Initial Read Channel Program

This is a representation of the initial read channel program. It is not a complete picture of all the CCWs, but is meant to illustrate the lock read and set operation. It begins by locating the lock record, which is the second record on the first track of the checkpoint. It then uses SEARCH-KEY-EQUAL to ensure that the lock is currently unowned. If the key of the lock record is currently zero, it then goes on to read the master record (record 3) and the check record (record 1). Finally it sets the lock by writing its system ID into the key and data of the lock record. This channel program normally ends at the NOP.

If the SEARCH-KEY-EQUAL operation fails because the lock record is currently non-zero, the channel program continues at the label FAIL which reads the value of the lock record in order to determine which processor owns the checkpoint lock.

When this situation occurs, JES2 will attempt its own error recovery: a warning message will be issued to the operator, and the operation will be periodically retried. If the system which lost the RESERVE is still running, it will eventually clear the key value, allowing the looping systems to proceed. If it is not running, a JES2 operator command will be used by one of the other systems to reset the lock value, on behalf of the failed member. Thus this lock can be used as a backup for the RESERVE/RELEASE hardware mechanism, since it tends to lock 'closed', rather than 'open', when hardware or software failures occur.

Extraordinary Error Detection and Recovery

Another type of recovery which has been implemented in this fix is recovery from all kinds of extraordinary errors. These can be characterized as errors which are not detectable with normal procedures. They are frequently caused by external events or actions. While these events are rare, they do occur. Among the sources are such events as power failures, hardware errors, and damage to the checkpoint media. Also, operator or maintenance controls such as the system-reset key or the channel-reset key have also been known to damage the checkpoint data set.

These types of external events can occur at any time. They can interrupt channel programs at unfortunate times. This may lead to checkpoint data sets in which some records have been updated and others have not, causing queues or other checkpoint structures to be scrambled or inconsistent. It is even possible for a single CCW, like a WRITE, to be interrupted in the middle of data transfer. This results in the remainder of the record being zeroed out.

With these changes it is now possible for JES2 to detect and recover from many of these extraordinary errors. Standard MVS error recovery procedures are invoked to retry temporary I/O errors, except when the error occurs in the initial lock/read channel program. When the extraordinary error does not involve a processor failure (red light), it is now possible for the processor initiating the checkpoint I/O to detect the failure and attempt recovery. When the error disables the processor which has initiated the I/O, it is now possible for another processor in the complex to detect the failure. Because these errors involve incomplete channel programs and data transfer, in cases when the processor initiating the I/O is disabled, the primary data set probably was destroyed, and recovery involves using the alternate checkpoint, if it is present. If no secondary checkpoint data set is available and the primary data set is known to be damaged, then the copy (downlevel) of the checkpoint data that is present in the detecting processor is written to the primary data set for use during a warm start.

In certain cases where multiple failures occur and no duplexing protection is operational, JES2 may not be able to provide a method of recovery.

Temporary errors are retried using standard MVS error recovery procedures in most cases. When errors are considered permanent by the ERPS, they will still be retried by JES2 a limited number of times. MVS ERPs can be invoked by JES2 for all checkpoint I/O except that involving the initial lock/read channel program. Since MVS ERPs attempt to restart channel programs with the failing CCW, they cannot be used for the channel program performing the locking operation since this could invalidate the lock.

The first step in trying to detect interrupted channel programs is to ensure that any I/O initiated by this processor completes. In order to determine whether our own channel programs have completed normally, each one terminates with a special verification operation. This consists of a READ COUNT CCW which reads a known count field of some record into an area of storage which has been initialized to all binary ones (X'FF'). Since all binary ones is not a legal record ID (count field), JES2 can now detect whether its checkpoint channel programs have completed by examining the content of this storage area. When incomplete channel programs are detected in this way, they are always restarted from the beginning.

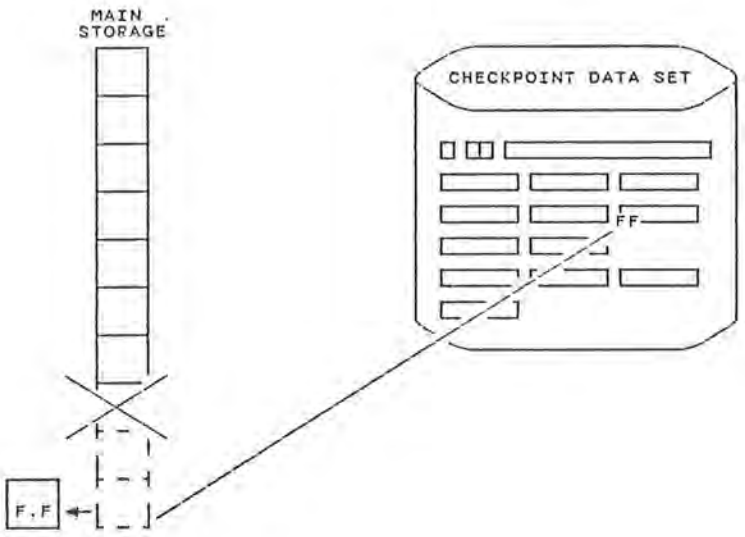

Figure 8. I/O Completion Verification

With these changes, JES2 will now also be able to detect when the checkpoint data read at the beginning of the checkpoint cycle is invalid. If another processor in the complex has gone down in the middle of a checkpoint update, the data on the primary checkpoint data set may have been partially updated and is therefore not valid. It is now possible to detect that this has occurred by using the check value in the first record of the checkpoint data set. Every time data is written to the checkpoint, an incremented counter (which ranges from 1 to 127) will be recorded in both the master checkpoint record and this check record. All checkpoint write operations will write the master record (with the stored value) first. Then, after all the changed queue records have been written, the check record will be written as the last write operation (just before the I/O completion verification). When a processor completes a read for the checkpoint information, it can compare these two values to determine the integrity of the checkpoint data. If the values are not equal (normally they would differ by one), then one of the other processors in the complex has failed during a checkpoint write operation and error recovery action is necessary, and always involves a warm start.

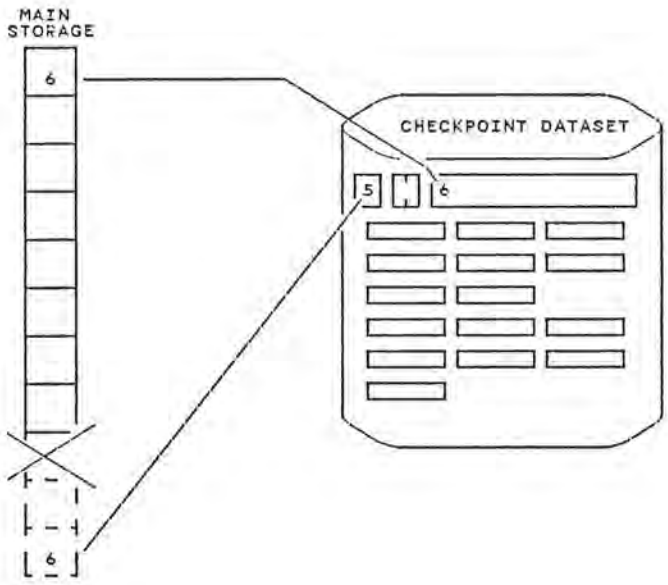

Figure 9. Checkpoint Data Integrity Verification

When JES2 is writing the checkpoint data set, either during normal operations or during error recovery, and a permanent error is detected, JES2 can be fairly certain that the copy of the checkpoint data it was writing is now destroyed. In these cases JES2 can reformat the data set during normal system operation using format-write CCWs. There are two situations under which this can occur: detection of an error during a write operation, and when the primary data set becomes unreadable and no duplex data set has been defined. In this case, we cannot assume that another processor can read the data set and it must therefore be completely rewritten, or there is a chance of losing the only checkpoint data set which exists. In both of these cases, the damaged data set is completely rewritten, using format-write CCWs to ensure that track condition errors like missing address markers or format errors created by interrupted channel programs are eliminated.

JES2 Initialization Checkpoint Data Area Analysis

In some cases, even after our best efforts to recover, the JES2 checkpoint may contain errors that are introduced either by unrecovered failures or software errors that may be too subtle to detect. In these examples, new processing in JES2 initialization will be called into play. This processing attempts, during complex-wide restarts, to analyze and correct errors in both the job and output queues.

The job queue and the job output table are scanned using their sequential organization (not dependent on chains), and the status of each element is determined. Chains pointing from one element to another within the structural organization of the queues are validity checked, and all elements are accounted for. If errors do exist, they are reported to the operator. In some cases erroneous chains and counters can be replaced or reconstructed, but in other cases the associated jobs or SYSOUT data sets will be purged from the queues. In the case of the job queue, the operator will be allowed to determine whether restart should continue if errors are found; but the analysis of the job output table will be performed and errors will be repaired without operator intervention. When this analysis is depended on, it is probably wise to make copies of the primary (or alternate) checkpoint data set prior to restarting JES2. This allows the diagnosis of the problem by examining the anomalies present in the unchanged checkpoint data set.

Diagnostic and Operational Considerations

As stated at the outset, a primary goal in making these changes was to permit simpler diagnosis of checkpoint problems that do occur. To this end, the JES2 checkpoint integrity software has eliminated the confusing combinations of catastrophic error messages and unexplained terminations that used to occur, and has replaced them with a set of error terminations (Kxx CATASTROPHIC ERRORS) that are unique to each error scenario. Each Kxx termination code has documentation (in the System Message Library) that describes the nature of the error situation; the specific effects of the failure, such as which of the checkpoint data sets is likely to be damaged; the actions the system has already taken and what the results of these actions were; and the actions that are appropriate as operator/system programmer responses to the failure. Also in all error scenarios a dump of the checkpoint data can be printed from the SYS1.DUMP data set that JES2 wrote at the time the error was detected.

In conjunction with the enhanced recovery options that these software changes provide, there two new operational characteristics of interest. First, the operator is now provided with an initialization option 'ALTCKPT', that will cause JES2 to read the alternate checkpoint data set as the source of checkpoint information during initialization. Secondly, an additional operand of the $ESYS command is provided to allow the operator of a processor to manually release the checkpoint lock that had been set by another processor that failed.

Also, a new initialization statement is available to specify the volume serial on which the alternate checkpoint data set is located.

Although these facilities are made available as operator functions and also provide significant internal safeguards against their misuse, it is recommended that operators be fully trained in how to respond to checkpoint failure situations.

The key to complete understanding of the new checkpoint integrity mechanism and correct responses during critical periods after a checkpoint failure is EDUCATION! Installations should establish procedures which must be carried out faithfully in these situations, based on the recommendations in the System Programming Library documentation for JES2. During the initial testing of these changes and subsequent early user experiences with these scenarios, operator and system programmer procedural errors were common causes of checkpoint loss. We recommend that operators be assisted by system programmers (at least initially) when confronted with these potential failure events. Operators must be impressed with the gravity of these situations and must not attempt recovery actions for which the consequences are not known to them. Misuse of system control and main-

tenance keys like SYSTEM RESET or PSW RESTART should be discouraged. Above all, the checkpoint data sets of JES2 should be treated as critical system resources, and decisions involving their future should not be made by a single individual.

THIS PAGE INTENTIONALLY LEFT BLANK

READER'S COMMENT FORM

Title: The JES2 Checkpoint Mechanism
Washington Systems Center
Technical Bulletin GG22-9220-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Please state your occupation:

Comments:

Please mail to: J.M.Hutchinson
IBM Washington System Center
18100 Frederick Pike
Gaithersburg, MD 20760

READER'S COMMENT FORM

Title: The JES2 Checkpoint Mechanism
Washington Systems Center
Technical Bulletin GG22-9220-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Please state your occupation:

Comments:

Please mail to: J.M.Hutchinson
IBM Washington System Center
18100 Frederick Pike
Gaithersburg, MD 20760

GG22-9220-00

The JES2 Checkpoint Mechanism Printed in U.S.A. GG22-9220-00

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601