Rapid Data Recovery

Roy Cornford
cornford@us.ibm.com

John Campbell
campbelj@uk.ibm.com

DB2 for z/OS Development,
IBM Silicon Valley Laboratory

Rapid Data Recovery

With increasing demands on availability and growing volumes of data, speed of recovery is ever more critical when things have gone wrong. This short paper looks at DB2 for z/OS object based recovery and factors that influence recovery times and recommendations on best practices.

The type of recovery being considered here involves many DB2 table spaces and can be characterised as one of the following:

- On-site total system recovery to the current or previous point in time (PIT)
- On-site partial recovery, subset of applications, to the current or previous PIT
- Traditional offsite recovery to last log record.

There are various reasons for needing to recover many table spaces, including:

- Hardware errors
- DB2 system software errors
- Other system software or microcode errors
- Application software errors
- Disaster recovery scenarios
- Operational errors.

To achieve a quick resolution to whatever error has occurred, installations must be prepared for the event. The key preparations are

- To know what tasks are required to perform a recovery
- Knowing in what order to perform the recovery tasks
- Having prepared the environment ahead of the error or failure.

At the time of writing, DB2 Version 8 is the latest generally available version of DB2 for z/OS. Most of the considerations apply equally to Version 7 as to Version 8. It will be indicated when Version 8 in particular is required.

Preparing for recovery

Knowing the recovery requirements are essential as this will drive amongst other things the backup strategy. An established service level agreement will provide goals during recovery testing and provide a basis for cost justification in establishing the recovery strategy.

The principle areas for reducing recovery times of essential data are:

1. Data independence across applications
2. Data archiving
3. Disk (DASD) volume backups, in addition to regular DB2 image copies
4. Frequency and types of image copy backups
5. Support of parallelism during recovery
6. Faster media for recovery (DASD versus tape devices).

Making data independent across applications

When applications share data it makes it harder to prioritise the recovery. A critical application could end up waiting on a generally less important application if it uses data owned by the latter. If the data is infrequently changed, a possible solution could be to have two copies of data and periodically resynchronise. The goal is to in effect put a “data fire wall” around the applications so they can be recovered independently and prioritised. By fencing data off in this fashion, it could also assist if one application is forced back to a previous point in time.

If data is shared across applications and it is not feasible to create this type of “data fire wall” then this fact should be recorded. With this knowledge suitable prioritisation of these specific tables can be done to ensure they are on the critical path at recovery time.

Data Archiving

The goal here is a separation of active and the often larger, inactive data (operational history data). If the application can be structured so that rows that are generally static or fixed can be separated from the volatile rows then we can potentially enhance recovery.

An example: If a table contains data relating to an ordering application then moving completed orders to a separate table (a “history” table) after a period of time, would significantly reduce the size of the main table and hence the recovery time. There could also be the opportunity to bring the application back quicker, but perhaps without some of the history data (reduced function mode).

Another option could be to separate data by time period, either by having separate tables for given time periods e.g. one table per month, or by partitioning a table by time. Note that table based partitioning in Version 8, means that the partitioning can be done by time, whilst clustering can be done by some other criteria. Increasing the

maximum number partitions in Version 8, along with, the ability to add additional partitions makes such tables more manageable.

Benefits of having the data split between two table spaces rather than one (one active, one history):

1. The restore phase of recovery, the copying of the image copy datasets back to the linear VSAM dataset, can be done in parallel

2. As previously indicated the history data recovery could be deferred, if acceptable to the application/ business. For example, an internet banking system may only display the very recent transactions in “reduced function mode”, as opposed to all transactions spanning many months

3. The history data ideally will have no log apply phase, as moving data to history or deleting data from history is likely to be a batch process and image copies could be scheduled immediately after such processes

4. There is potential for enhanced image copy frequency on the table space with the active data

5. The table space with active data is likely to benefit during log apply, as the changes will be more concentrated and require less I/O.

The last point relates to the applying of updates as encountered in the log scan when an update is found in the log the associated page must be in the buffer pool or else brought in from DASD. Having the active data spread over fewer pages increases the likelihood of other rows in the same page having been recently updated and the page still being in memory. There is also greater chance of multiple updates being included before the subsequent externalisation of the page to DASD. Fast Log Apply (FLA), which is discussed later, will be more efficient the more the updated pages are clustered together.

Volume backups, in addition to regular DB2 image copies

In addition to DB2 image copies, backup of data can be taken at the volume level, to facilitate system wide recoveries. This can exploit various DASD hardware facilities to achieve very fast copy times and corresponding restores. By using the -SET LOG SUSPEND/RESUME commands, activity can be halted and changes externalised to DASD to achieve a consistent copy of the application(s). Note this is a system wide option and the DB2 log datasets are copied in addition to the data. Recovery to the PIT of copy involves a DB2 conditional restart to resolve in flight units of recovery.

By using the volume level backups the equivalent restore part of the recover could be extremely fast.

Prior to Version 8 of DB2 using such backup to anything else than the point the copy was taken, involved either running RECOVER LOGONLY utilities against every table space or having to analyse which ones had changed since the backup and selectively doing these.

With Version 8 there are two important improvements; both in the way the backup can be taken and in how the recover is performed, both in operation simplicity and performance. Note these operate at the system level (or group level in data sharing).

Firstly, the backup can now be taken using the new SYSTEM BACKUP utility (requires z/0S 1.5), which is less disruptive than the –SET LOG SUSPEND/RESUME alternative. The SYSTEM BACKUP utility allows the data, or the logs and the data to be copied.

Secondly, there is a new SYSTEM RESTORE utility, for recovering the data to a given point in time. The SYSTEM RESTORE is capable of performing both the "restore" of the data and the log apply or just the log apply (so can be used even if not on z/OS 1.5 if the volume restore is done outside of DB2 utilities control). The SYSTEM RESTORE utility traverses the log once, triggering the updates to effected rows. The SYSTEM RESTORE utility employs processing equivalent to a DB2 restart so can handle incomplete units of recovery. Submitting one utility to recover an entire system greatly simplifies the process and reduces possible errors, and improves the recovery time.

Taking image copies

One factor in the length of recovery time is the amount of log data that needs to be scanned and the number of updates that need to be applied. This time will be reduced when an image copy was taken close to the time that we are trying to recovery to, be it the current point in time or a previous point in time. Taking frequent image copies can significantly cut the log apply time when recovering.

Image copies can be taken using SHRLEVEL CHANGE or REFERENCE. It is expected that SHRLEVEL CHANGE will be predominately used when high availability is required. If a recovery point is required, then the QUIESCE utility could be used to create such a point, although this can be quite difficult in highly active systems. For example, the logical end of the business day could include a quiesce of the application tables. Another example may be to quiesce at a significant point in processing like providing files to other applications or institutions.

The full image copy is usually the foundation of the backup strategy and can be the traditional DB2 variety (simply referred to as full image copies, throughout this paper) or a concurrent copy, an advantage of the former is that it allows for subsequent incremental image copies to be taken.

In establishing an image copy strategy the following should be considered:

1) Media to place the image copy (DASD versus tape/cartridge)

2) Types of image copy
   a) Full image copy, including inline copies.
   b) Incremental image copy
   c) Concurrent copy

3) Frequency of copy
   a) Importance of data
   b) Amount of change to data
   c) Performance of log apply
      i) Size of active logs
      ii) Archive log media

Following on from the above, it should be concluded that one size will not fit all, not only can strategies vary between installations, but between applications within the same DB2.

DASD versus cartridge for image copies

Cartridge or tape has two principle advantages; firstly as a removable media, it allows backups to be physically moved, for example for disaster recovery. Secondly it is relatively inexpensive. The disadvantages of using cartridges include slower access and multiple jobs not being able to access the volume simultaneously.

For best performance it is desirable to have the image copy on DASD, both for absolute speed and concurrency. Tape can reduce concurrency in two ways. Firstly the numbers of tape devices/ units are limited, so the number of concurrent tape accesses is limited. Secondly, when datasets are stacked on the tape, then only one dataset is accessible at a time. The cost can be reduced by initially writing to disk and allowing DFSMS & DFHSM to manage the image copies, migrating them to a cheaper media after either a period of time. When the copies are part of a GDG, the migration criteria can be set up so that the most recent copies remain on DASD. Obviously a too aggressive migration policy can mitigate some of the advantages of being on DASD.

Also it is important to ensure space is available to recall the image copies when need, especially during a mass recovery.

When image copies cannot be accommodated on DASD then consider dual copying (local backup) to reduce the likelihood of falling back to an earlier image copy (this would lead to additional log data processing). However, resist placing small image copies on cartridge as the cost of retrieval can be disproportionately expensive.

Whilst size is a criterion for determining candidate objects for DASD, importance of the data is also a factor. In particular aim to have the copies of the DB2 catalog and directory on DASD.

When there are dual copies remember the copies could be on different media, i.e. primary on DASD and local backup on tape. This applies not only when the COPY utility makes multiple copies, but also when the COPYTOCOPY utility is used to create additional copies.

DB2 will fallback to a prior image copy if the desired one is not usable. It is recommended that a second image copy be created when there is an unrecoverable event i.e. a LOAD or a REORG. The second copy could be a candidate for cartridge, either as an inline copy created by LOAD or REORG, a copy created by COPYTOCOPY or the COPY utility.

Supplement full image copy frequency with incremental image copies.

Incremental image copies can be much quicker, can put a lower pressure on buffer pool resources and produce smaller output datasets than full image copies. Generally incremental image copies are more efficient when less than 10% of the pages have been changed. Although, if the changes are clustered, for example appearing at the end of the table space, they can still be more efficient even when up to half the total number of pages in the table space have been changed.

The CHANGELIMIT option on COPY can be used to control whether image copies are taken and what type is produced e.g. CHANGELIMIT(5,50) means if 50% of the pages have changed then a full image copy will be taken. If less than 5% of pages have changed then no copy will be taken, otherwise an incremental is produced. Remember it is essential that a full image copy is created periodically. Also ensure GDG bases are big enough to ensure that any required full image copy is not rolled off.

The use of real time statistics (RTS) could be used to develop a more sophisticated policy for taking copies.

The presence of incremental image copy, since the last full image copy, can significantly reduce the time during the log apply phase of recover, however the restore phase will be longer as pages from the incremental must be merged with the full image copy. The more incremental copies that must be merged the greater the overhead at recover time. This additional processing should be done ahead of any recovery, by using the MERGECOPY utility.

Ideally incremental image copies would be directed to DASD, however if they do end up on tape, then it is very important to ensure that sufficient tape devices are available when the copies are required. If DB2 cannot allocate all the copies at recover time then one or more incremental copies will be ignored and additional log records will need to be scanned. The MERGECOPY utility can again be beneficial by either allowing a consolidated incremental image copy to be produced, hopefully on DASD, or even better, a new full image copy.

Frequency for taking image copies

As the reason for taking frequent image copies is to reduce the amount of log data that must processed, table spaces that have high rates of change activity should generally be copied more frequently, than more static ones. Table spaces with critical data should also have a higher copying frequency; the most critical of all is the DB2 catalog and directory, which could certainly justify being copied multiple times a day. One approach could be along the lines of characterising data into groups, say high medium and low and copy every six hours, every twelve hours and once a day respectively.

Should log data not be on DASD or kept there for a significant period, then a more aggressive copy frequency may be needed to maintain recovery goals.

It is important to draw the distinction between data written to DASD and then migrated to tape and data written and still residing on DASD. Once the data is migrated to tape, it acquires many of the disadvantages associated with tape, since it must be recalled to be usable. Similarly, data written to a virtual tape server (VTS) is still tape from an operating system perspective, this is particularly important from a concurrency perspective. The number of units available or paths is limited and datasets on the same volume will have access serialised.

So far we have concentrated on the data itself; there is now the option of also taking image copies of the indexes. Having image copies of the indexes can significantly improve overall recovery time, versus a recover of the data followed by a rebuild of the indexes. When the REBUILD utility is used to “recover” indexes, the underlying

data must have already been recovered. By using the RECOVER utility, the indexes can be recovered alongside the data; this is particularly beneficial for large non partitioned indexes. In fact it is advantageous to have one RECOVER utility process all the indexes and the data in one invocation. Where physical database corruption has occurred or database integrity may have been compromised, it is highly advisable to rebuild the indexes from the underlying data, rather from an image copy.

Performing the recover

The overall goal is to minimise the total elapsed time. The principle factors are:

1. Optimising speed of individual recoveries
2. Obtaining a high degree of parallelism
3. Prioritising recovery tasks.

We have already indicated the desirability of having a recent image copy and it being on DASD, ideally any incremental image copies would have been merged ahead of being needed. The image copy being on DASD will minimise the restore phase of the recover. The log apply phase, which comprises scanning the log and processing the updates, will also benefit from the log data being on DASD, especially if an active log. The next important performance feature to exploit is fast log apply (FLA), which should be enabled ahead of any recovery situation, as it also benefits start database commands to clear the LPL restrictive state. Recovery jobs using FLA have been measured at 4 times faster than the equivalent without FLA enabled.

Use of Fast Log Apply

Fast log apply is enabled at the subsystem level, controlled by the DSNZPARM LOGAPSTG. This parameter controls the amount of storage allocated to the fast log apply function in the DBM1 address space, the default differs between DB2 versions. In version 7 it is zero, i.e. FLA is not enabled, except for DB2 restart processing. It is recommended that FLA is enabled and the maximum storage of 100Mb be used (the default from V8). Note this storage is acquired by DB2 when needed and subsequently released. Customers with high virtual storage requirements are recommended to maintain 200Mb “head room”, that is not to allocate storage up to the MVS limit, but to allow a cushion of 200Mb. The FLA buffer can use this during periods when recovery is being underdone.

The SYSTEM RESTORE utility will use FLA, provided LOGAPSTG is not set to zero.

FLA greatly improves overall elapsed time and the retrieval of the pages that need to be changed in the object being recovered. This comes about partially by allowing the use of prefetch, rather than doing single, random I/O. There is also overlap between scanning the log and processing the changes. When FLA is used the LOG APPLY phase could be faster by ten times or more.

To maximise the benefit of FLA, the following should be born in mind. The DB2 RECOVER utility can process a list of objects (table spaces or index spaces) to be recovered and results in the log being traversed once. A list in this case is of the format:

```
RECOVER TABLESPACE MYDB.MYTS1
        TABLESPACE MYDB.MYTS2
```

FLA will work on the list; however it is most efficient when the list contains no more than 98 objects. If 99 or more objects appear in a single recovery list then FLA will still be used, but some serialisation will occur, limiting the benefit. Each invocation of RECOVER will acquire 10Mb of FLA storage, so up to 10 RECOVER jobs can run on a given subsystem and benefit from FLA (assuming maximum of 100Mb is specified for FLA). Once there are ten recoveries on the subsystem, then any further RECOVER jobs will run without FLA.

Log considerations

Dual logging should be in use, with each set of logs separated onto different DASD subsystems, where possible. Define large log datasets, 4 GB are now possible, as “extended format” datasets and allow VSAM stripping. The number of datasets will depend on the amount of data the installation has decided to keep on DASD and what media is being used for archives. Dual archiving should be performed, note that ARCHLOG2 could be to a different media than ARCHLOG1, for example, place ARHCLOG1 to DASD and ARCHLOG2 to cartridge.

If the archive logging is directly to tape then the number of tape units available for the archive logs can be adjusted via the –SET ARCHIVE LOG command. This command can also be used to control how long individual logs remain allocated. During a mass recovery increasing the availability of tape resources for the archives data can be desirable, although other demands on tape devices may also be important, especially if image copies have been written to tape.

Using large, dual active logs will reduce task switching and allow pre-fetching of the log control intervals. DB2 will load balance between LOGCOPY1 and LOGCOPY2 (active logs, not archive logs, even if on DASD).

DB2 version 8 allows more data to be held in the active log set, by increasing the number of datasets that can be allocated from 31 to 93.

Some reasons for requiring a recovery may not become immediately apparent, for example some logical data corruptions, perhaps caused by a rogue application and not accompanied by an ABEND. For this reason it is desirable to maintain log data on DASD, either as archive logs or even better if in active logs, for a reasonable period, it is suggested that 48 hours be a goal.

Whilst these benefits are not confined to the logging infrastructure the following features should be exploited when an IBM ESS DASD subsystem is being used:

1. Multiple Allegiance (MA), allows more than one MVS to send I/O requests to the same DASD volume at the same time

2. Parallel Access Volume (PAV), allows multiple I/O to the same volume, from the same MVS. WLM can be used to dynamically control the number of concurrent access to a logical volume

3. I/O Request Priority (IORP) permits queued I/O requests to be handled by priority.

The use of IBM DB2 Archive Log Accelerator for z/OS, Version 2, makes improvements regarding archive logging. The archive log datasets can be compressed by DFSMS, making it more feasible to keep the data on DASD. Reductions in elapsed time will be up to 30% with the use of this product.

Recovering in parallel

Parallelism in recovering DB2 objects can be achieved in two ways, the RECOVER utility can be instructed to do parallel recoveries and multiple recovery jobs can be submitted to run alongside one another.

The PARALLEL keyword on RECOVER means that the utility will perform the restore phase in parallel for a list of objects. By dispatching subtasks to perform the restores of individual objects from their image copies the total elapsed time will be reduced. The RECOVER utility has 49 parallel task pairs at it's disposal, if the list is greater than this parallelism still occurs, but some restores will have to wait on others finishing.

Although up to 49 parallel restores could be taking place, this is a maximum and will be reduced either on request (via NUM-OBJECTS keyword) or when DB2 calculates

that there is insufficient virtual storage (NUM-OBJECTS omitted or specified as zero). Importantly the degree of parallelism may be significantly reduced when image copies are on tape or cartridge devices. This can occur either because of the image copy datasets being stacked on the same volume (results in serialisation) or due to the number of tape units available. This is another advantage of the image copy being placed on DASD. PARALLEL can still be specified when there is a mix of tape and DASD image copies. PARALLEL is also an option on the COPY utility for minimising the time taken to produce the copies.

By specifying a list of objects to the RECOVER utility we can reduce the activity against the logs as the log is scanned only once for the entire list. By adding the PARALLEL keyword we can obtain parallel restores from the image copies. As has been noted it is important not to specify too long a list, it is suggested that about forty should be used as a rough upper limit, certainly going over 98 should be actively avoided. Putting objects with similar update patterns in the same list is also desirable, so that the same parts of the log are accessed, for example, all the parts from the same table space and the associated indexes.

The degree of parallelism that could be expected when image copies are predominantly on DASD is approximately 40 X 10 X n, where n is the number of members in the data sharing group. So a 4-way data sharing group could be handling approximately 1600 objects at one time, with a high degree of efficiency from a DB2 perspective. Note this will place very high demands on the I/O subsystem, possibly leading to contention.

If image copies are mainly on tape or cartridge, the degree of parallelism is going to be restricted by the number of tape units. The number of tape units to do restores of image copy datasets will be no more than the total number of units on the MVS system, less those dedicated to the archive log datasets, less other usage by the system, for example DFHSM. This number is going to be significantly less than the image copy on DASD equivalent of approximately 400 per member.

Prioritising the recover jobs

If the catalog and directory is required to be recovered then this must always be done first, ahead of application data. Refer to the current DB2 Administration Guide and Utility Guide and Reference, for procedure and sequence for recovering the catalog and directory, pay particular attention to recovering any user defined indexes that may exist.

Once recovery of the catalog and directory is complete, the application data can be recovered. If there is data independence between the applications, then prioritisation

could simply follow the relative importance of the applications involved. If table spaces are shared between applications then these should be recovered up front.

Advance knowledge of the relative time it takes to recover individual table spaces within an application, could allow for some critical path analysis, but this would have to have been established in advance. Assuming that there is not enough bandwidth to recover all objects in parallel, try to avoid having performed most of the recoveries and having a very lengthy single recovery at the end. Earlier we estimated 400 (40 objects in list X 10 jobs) objects per DB2 subsystem, considerably less for tape or cartridge. So some prioritisation would seem beneficial, particularly if tape units are a constraining factor. Consider giving priority to the following:

1. Larger table spaces; require long restore phase

2. Table spaces with known high update rates; require lots of log apply processing

3. Tables whose indexes don't have image copies; require the indexes to be rebuilt after the recovery.

4. Table spaces known to have regular updates; thereby having little opportunity to pseudo close (read-only switching) and consequently have long log scans.

With regards to the last point, it is recommended that the close rule for table spaces and index spaces, plus the DSNZPARM options of PCLOSEN and PCLOSET are reviewed to allow suitable read-only switching. Many installations have historically used a higher than needed PCLOSEN and PCLOSET values in order to avoid physically closing page sets. In doing so this has avoided the performance overhead associated with open/ close processing, but at the expense of recovery performance (longer log scans, as SYSLGRNX records remain open for longer periods). When assigning PCLOSEN and PCLOSET, plus setting CLOSE YES / NO, it should be appreciated that:

- Page sets with CLOSE NO will really not be closed at pseudo close. CLOSE NO page sets remain open indefinitely, being closed only when:

  - DB2 is stopped
  - A stop database command is issued
  - The maximum number of open datasets (DSMAX) is reached and all available page sets defined CLOSE YES have been closed

- Coupling facility (CF) name class queues from Version 7 improves the performance of transitioning to the non-group buffer pool dependant state: overhead of read-only switching reduced.

If the recovery of an index is being performed by rebuilding it, the REBUILD job can be done immediately after the recovery of the associated table space. The REBUILD can run alongside other recoveries, so whilst the number of RECOVER jobs should be limited to 10 per subsystem, there can be more than ten jobs involved in the recovery running at a time.

If any tables have been identified as being non essential, for example history or archive data, then these should be prioritised at a low level.

Summary

To achieve a high performance recovery, the environment must be analysed and a suitable strategy devised. The procedures must be routinely tested, perhaps every year or following the migration to a new version of DB2. The testing not only confirms the procedures work, but provides experience to the personnel responsible for the recovery. Experiences gained, especially recovery timings, should be used to adapt the procedures, particularly in sequencing recoveries. The procedures should also be periodically reviewed to incorporate new facilities in DB2 and/or other zSeries software or hardware.