Redpaper

Jeff Berger
Paolo Bruni

DB2 9 for z/OS: Backup and Recovery I/O Related Performance Considerations

Introduction

This IBM® Redpaper provides best practices and I/O-related performance considerations for backup and recovery using DB2® 9 for z/OS®. It describes some guidelines for DB2 for z/OS backup and recovery, then it discusses performance considerations that may be used to predict recovery time, which can be used, in turn, to adjust backup frequencies or methods. This Redpaper has the following sections:

- “Planning the backup strategy” on page 2

  This section deals with the series of decisions which a backup strategy involves, beginning on the day that disk and tape hardware are purchased. Configuring the DASD hardware is also the subject of Part 1. At this time some fundamental decisions are made that affects both the probability of a hardware failure and the amount of time that it takes you to recover from that failure. The decisions made at this time also affect the performance of online transaction workloads. Part 1 also explains and contrasts the two major types of backups, image copies and system level backups.

- “Recovery strategies” on page 16

  This section presents some recovery strategies for recovering the entire DB2 system to the current point in time after a hardware failure. This part explains the factors that affect the recovery time and how to predict the recovery time. Armed with the knowledge of how long recovery will take, it is easier to develop an effective backup strategy that enables you to achieve the recovery time objectives.

- “Backup strategies” on page 22

  This section analyzes hypothetical workload and failure scenarios: we examine a scenario of hardware failure and two scenarios involving programming errors. We examine related recovery to the current point in time and to a prior point in time.

This paper references and builds upon the IBM Redbooks® publication *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370, which focused on disaster recovery, but also introduced some concepts about FlashCopy® and system level backups. That book was written prior to

© Copyright IBM Corp. 2008. All rights reserved.

DB2 9 for z/OS, which added some new features that use FlashCopy and system level backups, and prior to the availability of the DS8000™. This paper contains a discussion of DB2 9 features and DS 8000 functions.

Other references include the following Redbooks publications:

- *IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786.

  This paper references a number of hardware concepts that are described in this Redpaper publication
- *IBM System Storage DS8000 Series: Copy Services with IBM System z,* SG24-6787,

  This Redbooks publication summarizes all of the FlashCopy capabilities of the DS8000 and z/OS
- *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation,* SG24-6789-02

  This Redbooks publication describes current tape drives and their performance.

Planning the backup strategy

The choices begin with the type and quantity of storage to purchase. You need to decide whether to use some form of data replication such as Metro Mirroring or Global Mirroring to protect against a disaster that may destroy the local computer center. You need to decide whether disks are used to store backups and archive logs and the number of disks to purchase. You need to decide whether to store backups on tapes and, if so, the type and quantity of tape subsystems to purchase.

Disk considerations

The DS8000 disk storage is the IBM solution for high-capacity, high-performance, mission-critical workloads of midrange to large enterprises. The DS8000 features many advanced management features, designed to enable and manage a tiered storage system. You can use Global Mirror, Metro Mirror, and Global Copy. In terms of capacity, the IBM DS8000 scales from 1.1 TB to over 320 TB in a single system. The IBM DS8000 supports heterogeneous storage across multiple applications and platforms with up to 128 4 Gbps Fibre Channel/FICON® ports or up to 64 IBM ESCON® ports.

The DS8000 disk systems can support up to 256 GB of cache to improve I/O efficiency. You can divide your DS8000 into two independent and unique storage environments that can run their own microcode and continue to support various heterogeneous workloads. These LPAR features maximize the efficiency, flexibility, and cost-effectiveness of your disk system.

IBM System Storage™ DS8000 (see Figure 1) supports both RAID 5 and RAID 10. Each RAID rank consists of eight disks.

Archived

Figure 1 DS8000

In a DS8000, a RAID 5 array built on one array site contains either seven or eight disks depending on whether the array site is supplying a spare. See Figure 2. A seven-disk array effectively uses one disk for parity, so it is referred to as a 6+P array (where P stands for parity). See Figure 2. The reason only seven disks are available to a 6+P array is that the eighth disk in the array site used to build the array is a spare. We then refer to this as a 6+P+S array site (where S stands for spare). An 8-disk array also effectively uses 1 disk for parity, so it is referred to as a 7+P array.

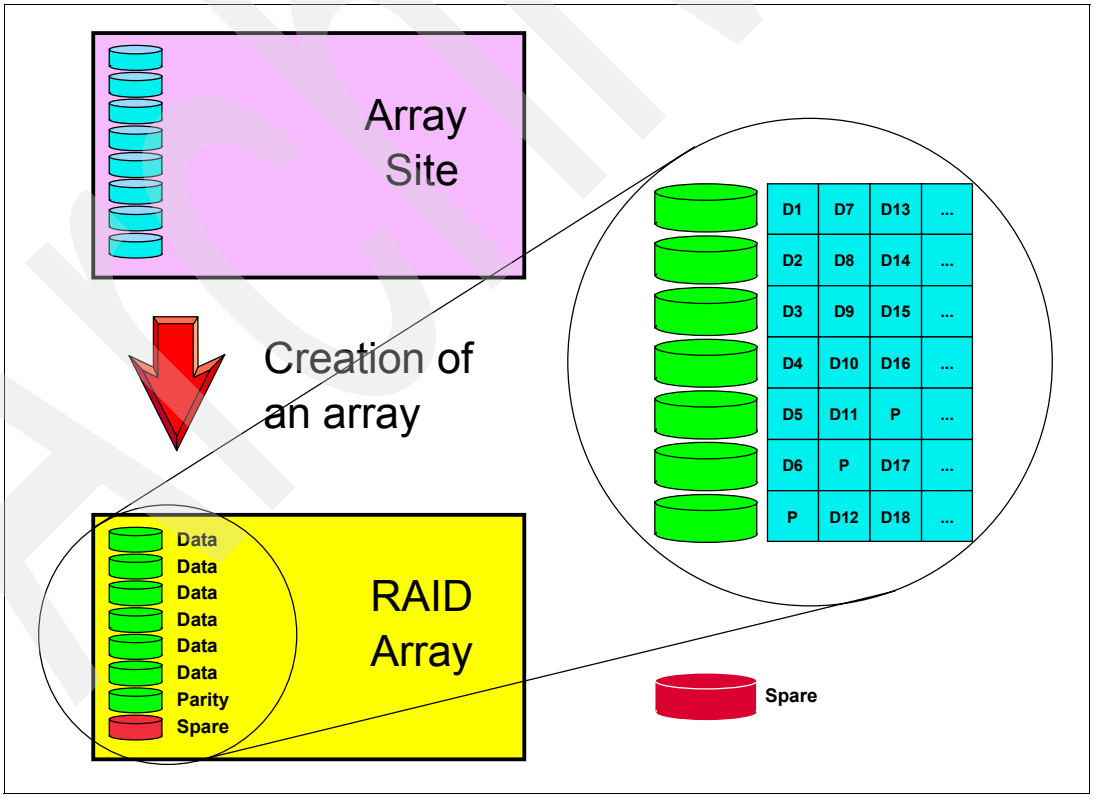

Figure 2 RAID Array

The data is striped across all of the disks except the spare. Whenever one disk in a rank fails, the parity is used to reconstruct the lost data to one of the spare disks. While the disk is being

reconstructed, performance may be degraded somewhat. The DS8000 slows down the rebuild process to minimize impact on host workload. The probability of two disks failing in the same rank is small, but that is one reason for taking backups. With RAID 5, if two out of eight disks (on a 7+P rank) or two out of seven disks (on a 6+P rank) fail, then the rank is said to have failed and you need to recover the lost data using backups. RAID 10 stripes the data across 4 disks and mirrors the data across the other 4 disks in the rank. Thus, 50% of the space is used for mirroring. However, reliability is better because recovery from the backups only need be done if two out of two disks fail, a lower probability than two out of six disks. RAID 10 also experiences better write performance than RAID 5 in case of intensive random updates. However, the sequential I/O performance of RAID 10 is not as good as RAID 5 because the data is striped across fewer disks. RAID 10 is not as commonly used as RAID 5, mainly because more raw disk capacity is needed for every GB of effective capacity.

When the DS8000 is configured, you decide how to partition the ranks into extent pools. An extent pool is a logical construct to aggregate the extents from a set of ranks to form a domain for extent allocation to a logical volume. Typically the set of ranks in the extent pool should have the same RAID type and the same disk RPM characteristics so that the extents in the extent pool have homogeneous characteristics.

Each RAID rank is assigned to an extent pool. Thus, your DB2 database and backups may be isolated from each other in separate extent pools so that no rank failure causes the loss of both the database and the backup. However, this reduces the performance of the transaction workloads because the transactions have accessibility to fewer disks.

The DS8000 now supports space-efficient (SE) FlashCopy[1], but DFSMShsm™ does not. Because DB2 is dependent on DFSMShsm support for FlashCopy, this function is not available.

After having configured the disk hardware, the DFSMS™ storage groups must be defined, perhaps with one storage group for incremental copies and archive logs, and a different storage group for system-level backups. If system-level backups are to be used, two copy pool backup storage groups should be created respectively for DB2 databases and logs. If these backups and archive logs are to be used to recover from a hardware failure, it is necessary to keep them physically separate from the database, with the knowledge that doing so adversely effects the performance level of the transaction workloads to some degree. You need to decide whether to use DFSMShsm to migrate the image copies and archive logs from disk to tape. If a lot of migrated data sets have to be recalled from tape during recovery, HSM and the tape subsystem may experience contention.

Tape considerations

If you decide to store backups on tape, you have the option whether to use low or high-capacity high-performance tapes. Low capacity tape cartridges should be avoided, because they are slow. Likewise, avoid using ESCON for tape because they are slow. Therefore, it is suggested to avoid low capacity tapes and avoid ESCON for tape.

You need to define the number of tape controllers and the number of tape drives. The number of tape controllers affects the aggregate bandwidth of the tape subsystems. For example, if one 3592 tape controller[2] has four tape drives (see Figure 3), only three can effectively transfer data at the same time, while the fourth tape drive is mounting or rewinding tapes.

[1] Space-efficient FlashCopy allows you to allocate only the physical space that you need to maintain a point-in-time image, rather than having to allocate the full amount of physical space on a volume. Space-efficient FlashCopy does not support data-set-level FlashCopy.

[2] The IBM TotalStorage® 3592 Enterprise Tape Drive has been replaced by the IBM System Storage TS1120 Tape Drive.

Figure 3 3592 J1A tape drive and 3592 J70 controller

Tape performance of modern FICON tape subsystems is good, but more than one tape controller may be necessary to achieve high bandwidth to support the recovery performance requirements. Even with high-performance high-capacity tapes, recovery performance suffers if only a small amount of data is stored on each tape because it takes tens of seconds to mount and rewind tapes.

Yet the 3592 architecture can be costly if the tapes are not fully utilized. One vendor advertises three size 3592 IBM cartridges that hold 60 GB, 300 GB, and 700 GB, costing $106, $133, and $180 respectively. The 3592 tape controller is capable of writing 256 KB blocks at 155 MBps. or more. So, if only 3 GB were written to tape that had 300 GB capacity, not only would 99% of the space be wasted, but it would take more of time to mount and demount the tapes than to write the data.

Therefore, use 3592 tapes to achieve optimal tape performance and utilize as much of the space on these tapes as possible to achieve the best performance at the lowest cost.

Tapes are ideal for storing full image copies, particularly when the image copies are stacked on tape by the COPY utility. Inline copies created by the LOAD or REORG utilities are better left on disk, unless several objects are REORGed at the same time using one REORG statement along with a TEMPLATE statement. COPY should not be used to store incremental copies on tape because it increases the number of tape drives needed for recovery without saving much disk space.

If you plan to store full image copies on tape, it is generally better to let COPY write the image copies of large objects directly to tape (using LISTDEF and tape stacking) rather than letting DFSMShsm (abbreviated here as HSM) migrate them to tape, even though COPY may take longer writing to tape than to disk. However, this strategy only makes sense if you ensure that COPY fully utilizes the tapes. The main reason for preferring COPY over HSM for tape management is to enable the RECOVER utility to minimize tape mounts[1]. This also makes it possible for DB2 to store the image copies using 256 KB block sizes. Some tape drives are faster than disks today for single threaded operations. If parallelism is used, offloading the write I/Os to a tape subsystem has the effect of relieving the load on the disk subsystem, resulting in better performance for online transactions.

You need to decide whether to purchase an automated tape library subsystem or a virtual tape subsystem. Consider that virtual tape subsystems can write the data quickly, but may have difficulty recovering the data because of an insufficient number of tape drives.

The 3592 tape controller is the best choice for tapes among IBM products, but Virtual Tape Subsystems (VTS) with FICON (and fiber connections between the disk cache and the tape drives) is a viable option if used appropriately. Even though VTS has a disk cache, it is known to z/OS as tape device and it has all the serialization characteristics of all tape devices. That

[1] HSM archives randomly and HRECALL is needed to make the data sets available preventively.

is, no two jobs can allocate the same tape at the same time. Such issues do not exist for disk volumes or data sets. Consider that if two DB2 members try to store image copies on the same tape, the second job waits for the tape to be made available.

HSM can be used for migrating the inline copies and incremental copies to tape, but HSM can be overwhelmed if a massive number of image copy data sets need to be recalled from tape. So, using HSM to migrate data sets to tape has dangerous implications for recovery performance. If you are concerned about incremental copies consuming disk space, rather than using HSM it would be better to write fresh full image copies to tape, although doing so increases the tape requirements. Remember that any time that the RECOVER utility spends waiting for HSM recalls offsets the advantage of having created the incremental copies—that is, to reduce the number of log records to be applied. In other words, if there is not enough disk space to store the incremental copies, then it may be better to simply increase the interval between successive COPY invocations and avoid incremental copies altogether. Therefore, store incremental copies on disk and avoid letting DFSMShsm migrate them to tape.

A good backup strategy avoids having to read archive logs during recovery, except as a last resort. Sometimes reading archive logs is unavoidable, however. DB2 supports the writing of archive logs directly to tape, but doing so is discouraged. Outside of the DB2 environment, it is possible to consolidate the archive log data sets to reduce the number of tapes and manually copy the data sets back to disk before attempting a recovery, but this is a manual task that is outside the scope of DB2. The preferred method of managing archive log data sets is to use HSM. To minimize the HSM recall time, it is a good idea to anticipate which archive logs will be needed and manually use the HRECALL command to queue the recalls together. That way, if HSM stored some of the archive log data sets on the same tape, it will recall all of the required data sets without having to remount the same tape. Make sure that there is enough disk space to hold all of the recalled data sets. Therefore, if archive logs are allowed to be migrated to tape by DFSMShsm, and if recovery needs to read the archive logs, use the HRECALL command to recall the data sets all together.

Optionally the recover jobs may be started without waiting for the archive log recalls to complete.

Data mirroring

A variety of techniques exist for replicating, backing up, and recovering data. The commonly accepted approach to backup and recovery is to mirror the data to a secondary disk storage system. This prevents one storage system from becoming a single point of failure. If the second system is remote, there is protection against disasters that may destroy the local site. Using a secondary storage system, however, may be the most expensive solution, whether or not the secondary server is at a remote site. Mirroring only protects against physical data corruption. It does not protect against logical data corruption, because the corruption just gets mirrored to the remote site. So, data mirroring must be combined with a point-in-time backup strategy. Data mirroring is covered extensively in the Redbooks publication *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370.

System-level backups

The BACKUP SYSTEM and RESTORE SYSTEM utilities were introduced in DB2 Version 8. The BACKUP SYSTEM utility is a method for creating a system-level point-in-time backup for both database and log copypools (option FULL) or only the data (option DATA ONLY). The RESTORE SYSTEM utility is used to restore the database copypool and apply the current log, or just apply the log (option LOG ONLY). The BACKUP and RESTORE SYSTEM utilities are described in the Redbooks publication *DB2 Version 9.1 for z/OS Utility Guide and*

Reference, SC18-9855. The IBM Redbooks publication *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370 shows some recovery scenarios. This book was written prior to the existence of DB2 9, however, and prior to the hardware support for incremental system level backups. The BACKUP SYSTEM utility uses FlashCopy, which is best described in the Redbooks publication *IBM System Storage DS8000 Series: Copy Services with System z*, SG24-6787. The DB2 BACKUP SYSTEM and RESTORE SYSTEM utilities use HSM to invoke FlashCopy services. In DB2 9 the RECOVER utility can also use HSM to invoke FlashCopy services. More about DFSMShsm can be found in the Redbooks publication *DFSMShsm Fast Replication Technical Guide*. SG24-7069. Also the DB2 CHECK utility may call DFSMSdss™ (known as DSS) to invoke FlashCopy services.

Two distinct methods of using the BACKUP SYSTEM utility are described in this section (both new with DB2 9 for z/OS):

- Use the system level backup merely as a transient means of dumping the database to tape. There is no intent to restore the database directly from disk.
- Maintain a permanent system level backup on disk and increments that backup periodically.

As we shall see, there are reasons why it is undesirable to combine these two methods. The first method is the cheaper of the two, but recovery is from tape rather than disk. The second method requires more disk space, but it allows recovery from a disk backup when there is a media failure.

With either method, if there is ever a need to recover to a time prior to the last increment, the data has to be restored from tape.

System level backups use DFSMS copy pool backup storage groups. A system level backup is also sometimes referred to as a Point-In-Time backup. The SMS configuration must be modified to enable system level backups. First, an SMS copy pool is defined for the DB2 database (named DSN$db2loc$DB where “db2loc” is the DB2 location name). The copy pool may contain up to 256 SMS storage group names. Then, one or more SMS storage groups with type “copy pool backup” are defined.

Note: The normal database storage groups use type “pool”.

For each storage group in the Copy Pool, a copy pool backup storage group is identified to SMS.

In order for the system level backup to protect against media failures, the copy pool and its backup need to reside on completely separate media within the same DS8100 storage server. If a DS8300 storage server is used, the copy pool and its backup need to reside in the same storage facility image (SFI), because FlashCopy does not allow the source and target on different SFIs. To reside in separate media means that the volumes in the storage group need to be allocated from different DS8000 extent pools. See Figure 4. Separating the copy pool from its backup into different logical subsystems (LSSs) is insufficient, unless the LSSs are allocated from different extent pools.

Archived

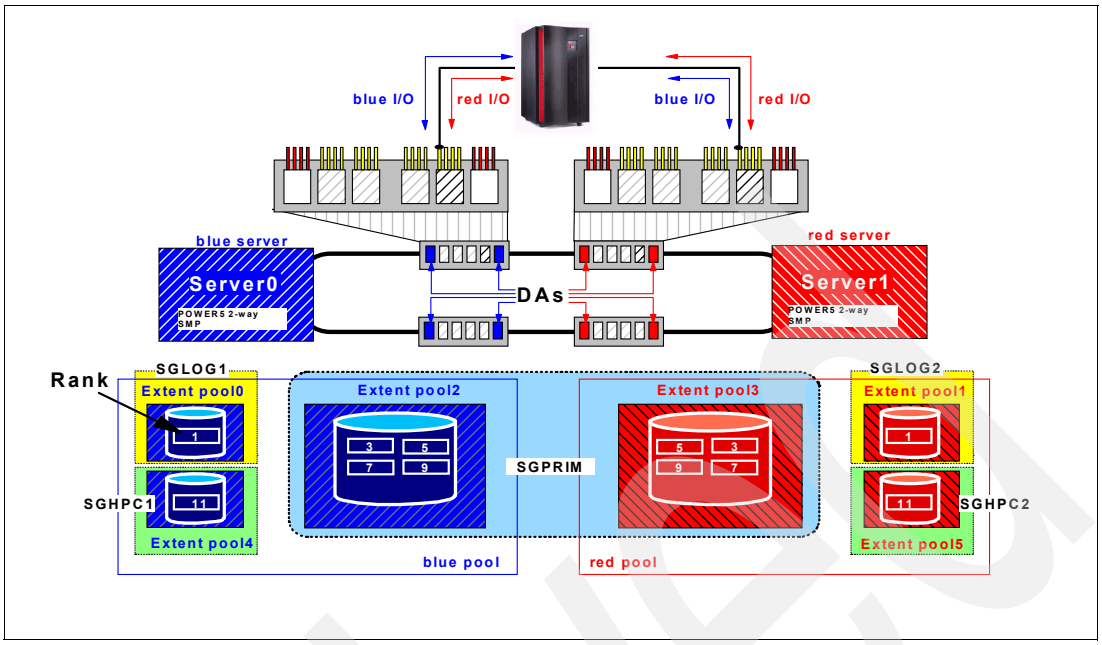

Figure 4 Extent pools and servers

Copy pools are discussed in Redbooks publication *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370. This book suggests at least two versions of the Copy Pool on disk (in addition to any copies that might be dumped to tape). Here, however, we recommend keeping no more than one version on disk because the DS8000 supports only one persistent FlashCopy target for a particular source. A persistent copy can be updated incrementally. The reason for changing the recommendation is because the incremental copy feature didn't exist when SG24-6370 was written. Two disk versions can be maintained, but creating fresh full volume backups frequently causes a lot of performance overhead. Because updating a single disk version incrementally and frequently means that you cannot use the disk version to recover to a prior point in time, use a tape backup to recover to a prior point in time.

It is useful to introduce some FlashCopy terminology to describe two flavors of FlashCopy, as determined by the COPY or NOCOPY option. Once a Flash Copy relationship is established, the COPY option initiates a physical background copy operation, but NOCOPY does not. Therefore, if the backup is intended as a means to recover from media failures, NOCOPY cannot be used, and the backup and database need to reside on different media. Until the background copy is complete (which may take several hours) the background copy is not secure from database media failures. Although the source volumes can be modified while the background copy operations are occurring, the target backup is not secure until the background copy is complete.

DB2 9 added the DUMP option to the BACKUP SYSTEM utility. This option causes HSM to dump the disk backup to tape. If VERSIONS=0, HSM uses the NOCOPY option. VERSIONS=0 offers better database performance for two reasons:

- The database and backup volumes can coexist on the same media, so that the database has access to all of the media, which is better for transaction performance.
- The NOCOPY has better performance than the COPY option, because only the updated tracks is physically copied.

If the dumps are deferred, however, the COPY option provides an opportunity for the DS8000 to copy the source tracks prior to the dump, and the dump task has less impact on the media that contains the source tracks.

HSM does not allow the database to be restored from the disk backup when VERSIONS=0. Even if HSM did allow it, if there were a database media failure, the disk version would be lost because HSM used the NOCOPY option. VERSIONS=0 is intended to be used just for dumping the backup to tape. The dump can also be deferred until later. If VERSIONS>0, HSM uses the COPY option which allows the database to be restored from a disk version.

Another addition to DB2 9 is the incremental capability. BACKUP SYSTEM ESTABLISH FINCREMENTAL specifies that a persistent FlashCopy relationship is to be established, if none exists, for the source copy volumes in the database copy pool. Use this keyword once to establish the persistent incremental FlashCopy relationships. Subsequent applications of BACKUP SYSTEM (without the FESTABLISH INCREMENTAL keywords) automatically processes the persistent incremental FlashCopy relationship. BACKUP SYSTEM END FINCREMENTAL specifies that a last incremental FlashCopy be taken, and for the persistent incremental FlashCopy relationship to be withdrawn for all volumes in the database copy pool.

Unlike traditional image copies, an incremental system level FlashCopy backup always overwrite any writes previously done to the backup. Thus, it is easy to maintain a merged system level backup. Such increments, however, preclude you from restoring to a prior point in time using that same backup version.

To achieve the best recovery time after a media failure with the least impact on workload performance when taking backups, use VERSIONS=1 and incremental backups. Perform frequent incremental updates to keep your system-level backup on disk current. Periodically dump the backup to tape, preferably during a time when the database I/O activity is low. Avoid ending the persistent FlashCopy relationships. To recover to prior points in time, you need to restore the backups from tape.

Arguably, the incremental update feature of the BACKUP SYSTEM utility is what makes system level backups so much superior to traditional image copies, because the cost of creating traditional incremental image copies is much higher. While it is true that FlashCopy avoids using channel resources and avoids using the DB2 buffer pool, the amount of background data movement is the same, and database availability is the same as the COPY utility with SHRLEVEL CHANGE.

If there is a database media failure before the dump is completed, then there is a need to use an older backup to recover the database. If the workload is busy while the DS8000 is trying to copy the data in the background, it may take several hours to complete the background copy of a large copy pool. During that time the database is exposed unless there is an older backup. To be 100% bullet proof, then, there is a need to have at least two backups at any given time. To store two backups on disk the storage requirements must be tripled. The exposure can be minimized, however, by taking frequent incremental backups and occasionally dumping the backup to tape. In this case, the database needs to be restored from tape only if there was a media failure while the system level backup was being incremented. Otherwise, the database can be restored from disk.

Copy pool versioning was designed for non-persistent backups. If you use VERSIONS=2 and one version is persistent, the BACKUP invocations bounce between the persistent and non-persistent backup, which is likely not the intent. Remember that full backups impacts performance much more than an incremental refresh of a backup.

Therefore, avoid using VERSIONS>1, because a volume can have only one persistent FlashCopy relationship.

Normally you use the RESTORE SYSTEM utility to recover from a system level backup to the current point in time. The RESTORE SYSTEM utility does not have the ability to recover different objects to different points in time. The RECOVER utility has that ability. In DB2 9, the

RECOVER utility was enhanced to recover individual DB2 objects from a system level backup in conjunction with changes to DFSMShsm Versions 1.8. This is typically what is done when there is a need to recover from a programming error and part of the database needs to be recovered to a prior point in time. If the system level backup on disk is only used for this purpose, then it is unnecessary to divide the database and backup on separate media. Backup can still be dumped to tape in order to protect against the rare media failures.

Use the RESTORE SYSTEM utility to restore the entire system to a particular point in time. Use the RECOVER utility if it is necessary to recover only a subset of objects, or if it is necessary to recover different objects to different points in time.

System level backups must be supplemented with inline image copies. After running the RESTORE SYSTEM utility, objects that had been loaded or reorganized because the system backup was created or last incremented are left in RECOVER PENDING state. Those objects have to be recovered from inline image copies using the RECOVER utility.

To minimize the likelihood of having to recover from inline copies, after reorganizing many tablespaces, increment the system level backup.

There are also some special system level backup considerations when using NOT LOGGED objects. See “NOT LOGGED” on page 14.

Image copies

Let’s see how the DB2 utilities are used to create and manage image copy data sets.

We begin with the COPY utility. The COPY utility can be used to create both full image copies and incremental copies. The COPY utility is the best method for writing a lot of image copies to tape. The primary reason that it is the best method is that it enables control of how the image copies are grouped on tape, organizing the data sets by *recovery group*. A recovery group is a set of objects that is recovered at the same time. This set of objects could be associated to a DB2 database for ease of operations. In contrast, the dumps created by the BACKUP SYSTEM utility organize the data sets on tape by volume. When there is a need to recover the entire system, the tape organization doesn’t matter, but if there is a need to recover only a subset of objects, or if there is a need to stage the recovery one set of objects at a time, the subset should be organized on a minimum number of tapes to optimize the recovery time. In addition, DB2 9 supports the ability to write 256 KB blocks to tape for better performance.

Given a list of objects, the COPY utility can use parallelism to write the image copies to disk or tape. The TAPEUNITS parameter is used to control how many tapes are used in parallel. When using tape parallelism, the COPY utility sorts the objects by size and distributes the objects on tape according to descending size, so that the amount of data stored on each tape tends to be uniform.

To get the best performance when recovering a subset of objects from tape, it is better to use the COPY utility to organize the image copies on tape by recovery group, rather than dumping a system level backup to tape. When doing so, use the TAPEUNITS and PARALLEL keywords for optimal COPY performance.

If you have enough disk space to store the small image copies on disk, then only store the larger image copies on tape. That could mean that the image copies for LOB and XML tablespaces and partitioned tablespaces are stored on tape, while other image copies are stored in disk.

The image copies of compressed tablespaces are compressed, but the image copies of compressed indexes are not compressed. In contrast, the index backups produced by the BACKUP SYSTEM utility are compressed if the source index is compressed.

Beginning in DB2 9, image copies can be written with a block size up to 256 KB for greater tape utilization and performance. Using a 256 KB instead of 28 KB increases the throughput to read and write large image copy data sets by 30% to 50%. Therefore, use 256 KB block sizes for tape image copies.

SHRLEVEL CHANGE can be used to allow online work to progress while COPY is executing. However, SHRLEVEL CHANGE does not produce a *consistent copy*. A consistent copy is one which contains all updates for committed transactions and no updates for uncommitted transactions. Regardless, RECOVER to current produces a consistent copy. In DB2 9 RECOVER to a point in time also produces a consistent copy. If it is desired that the image copy itself be consistent, then SHRLEVEL NONE or SHRLEVEL REFERENCE must be used, but that blocks applications from updating the table space. Therefore, use SHRLEVEL CHANGE for best database availability.

By specifying COPY INCREMENTAL, the COPY utility produces an incremental copy. DB2 reads the space map pages to determine which pages have been changed since the last COPY. DB2 then uses List Prefetch to read just those pages. Only the changed pages are written to the incremental image copy data set. Therefore, do not use incremental copy for small objects.

If image copies are used to restore the database, the RECOVER utility opens all incremental copy data sets at the same as the full image copy in order to merge them. Performance problems can arise if the incrementals are on different tapes; these problems can be severe if many objects are recovered in parallel. The same problems can occur if DFSMShsm is used to migrate the image copies to tape.

Store incremental image copies on disk. The tape subsystem may experience thrashing during recovery if DFSMShsm has migrated the image copies to tape. If there isn't enough disk space to maintain the old incremental copies on disk, it is better to delete them from the DB2 catalog than to let DFSMShsm migrate them to tape.

Occasionally it may become desirable to merge the incremental data sets using the MERGECOPY utility. There are two types of merges:

- NEWCOPY YES

  NEWCOPY YES produces a new full image copy. It also preserves the previous incremental copies in SYSCOPY.

- NEWCOPY NO.

  NEWCOPY NO only merges the incremental copies. It deletes the old incremental data sets and removes them from SYSCOPY.

  If recovery might be to a point in time prior to running MERGECOPY, it would be better to avoid NEWCOPY NO.

As a general rule, if the aggregate size of all of your incremental copies reaches 20% of the size of your full image copy, then generate fresh full image copies, unless the large size of the incremental copies is due to growth in the database object itself.

The COPY utility was changed in DB2 9 to use Most Recently Used (MRU) for the buffer pool in order to minimize the impact on the buffer pool. Thus, COPY in DB2 9 tends to have a smaller impact on the performance of a concurrent transaction workload. Unlike the COPY utility, MERGECOPY does not use the DB2 buffer pool. Like the REORG utility,

MERGECOPY can stack the image copy data sets on tape and it does not support parallelism.

To avoid any COPY PENDING conditions, the LOAD and REORG utilities can produce inline copies. Use SHRLEVEL CHANGE to allow concurrent updates. Inline copies are consistent. If REORG is given a list of objects to REORG at the same time, the utility can stack the image copies on tape, but REORG cannot distribute the inline copies to tape in the same way that the COPY utility does.

Combining and comparing system level backups with image copies on tape

Ordinarily, there is no reason to write both image copies and a system level backup to tape. A reasonable option, however, is to use VERSIONS=1 for your system level backup without dumping the backup to tape and use the COPY utility to write image copies to tape. If this combination of backups is chosen, the RESTORE SYSTEM utility can be used to recover the whole system to the current point in time, and the RECOVER utility can be used to recover any subset of the system from tape, or to recover the whole system to a time prior to the last system level backup.

If you are fortunate enough to never have to recover some objects from tape, then it doesn't matter which utility is used to write the tapes. It also doesn't much matter if you have to recover the entire system from tape. However, let's consider what events would require restoring from tape. Remember that if incremental system level backups are used, the backup is no good if there is a database media failure while the increment is being written. This occurrence may be unlikely, but it is one of the reasons why backups are dumped to tape. A second reason for dumping the backup to tape is so that you might be able to recover to a prior point in time.

If the entire system is to be recovered at the same time, then the RESTORE SYSTEM utility is the easiest means to do it. However, if some applications are a higher priority than other applications, then it may be best to stage the recovery one application at a time. In this case, the RECOVER utility should be used to manage each stage. The RECOVER utility performs better if the recovery group is stored on a minimum number of tapes, as organized by the COPY utility.

If it is desirable to stage the recovery because some applications are more important than others, then use the COPY utility to write image copies to tape and organize the data by application.

Now let's suppose that the application or programmer has corrupted or damaged the database in some fashion, and suppose that the damage affects only certain applications. If the entire database were recovered, some of the applications would become unavailable. Therefore, it is important to recover only the data that was necessary for the applications that were broken.

In order to distinguish between different applications and recover from situations where only some applications are broken as a result of damaged data, then use the COPY utility to write image copies to tape and organize the data by application.

To illustrate how COPY/RECOVER compares to BACKUP/RECOVER when storing backups on tape, let's consider an example. Suppose that 10,240 database objects were distributed across 512 volumes, and suppose that the 512 volumes can be dumped to 100 tapes. Now suppose that a group of 100 objects were corrupted. If COPY had been used to dump all 100 objects to the same tape, then RECOVER only needs to access that one tape. On the other hand, if these 100 objects were distributed across the 512 volumes, it is likely that to use the system level backup, RECOVER would have to access all 100 tapes, and it would have to

search for each object on tape rather than reading the objects in the order that they were stored on tape. The same thing could happen if DFSMShsm were used to archive the objects on tape.

DB2 logs

DB2 Version 8 supports up to 93 active log data sets of 4 GB each, offering up to a total of 372 GB of active log data sets. For best recover performance, use enough active log space so that when there is a need to restore from the most current image copy or system level backup, DB2 does not need to access any archive log data sets.

Therefore, it is urged to allocate enough space for the active log space so that when there is a need to restore from the most current image copy or system level backup, DB2 does not need to access any archive log data sets.

Updating intensive workloads may experience disk logging performance issues. In these cases, if the environment supports MIDAWs, you might consider striping your DB2 logs. High speed channels reduce the need to use a lot of stripes and using too many stripes sometimes has an adverse affect on log performance.

If you plan to stripe your logs for speed, as a general rule use no more than 4 stripes for your active logs with a 2 gigabit/second (gbps) channels and no more than 2 stripes with 4 gbps channels.

DB2 9 introduced support for extended format archive log data sets. Thus, archive logs can now be striped and compressed by DFSMS. Striping the archive logs is useful if the active logs are also striped.

If recovery needs to access archive log data sets, then there are concerns about where the archive log data sets are stored and the type of data sets that are used. When the archive logs are stored on disk, it is best to use DFSMS compression to achieve the best I/O performance. Striping without using compression does not improve read performance because DB2 9 reads only one block per read I/O, but the Basic Sequential Access Method (BSAM) overcomes DB2's buffer limitations for compressed data sets.

If using DB2 9, use DFSMS striping and compression for your archive logs, and use the same number of stripes as are used for active logs.

Try to keep enough archive logs on disk so that if there is a media failure, archive logs do not have to be fetched from tape. This means that any archive logs written since the last system level backup was created or incremented, or since the last image copies were created, should be on disk.

LISTDEF

The LISTDEF utility enables database objects to be grouped in reusable lists. These lists can be used in other utility control statements to indicate that the utility is to process all of the items in the list.

The LISTDEF statement can be used to standardize object lists and the utility control statements that refer to them. Standardization reduces the need to customize or alter utility job streams. If predefined lists are not used, the entire list of objects must be repeated on each utility control statement that processes the same group of objects.

Archived

LISTDEF makes it easy to coordinate recovery groups between the COPY and RECOVER utilities can use the same LISTDEF definition. The utilities can even retrieve a LISTDEF from a commonly shared sequential file.

LISTDEF uses INCLUDE and EXCLUDE keywords to construct the list. Objects of a specific type, such as LOBs, XMLs, and indexes, may be included. Wild card characters (*) and pattern matching characters (%) can be used. For example, you could specify DB* to select all database names that start with the letters DB. Objects may also be excluded from a list. For example, a list may consist of all tablespaces except for XYZ*.

Implicitly created tables use system-generated names for the table space and database. Because COPY and RECOVER do not directly support table names, LISTDEF is the only way to specify the table names for COPY and RECOVER to use without first searching the catalog to determine the system generated names.

NOT LOGGED

It's important to treat NOT LOGGED objects with extra care when recovering databases, especially if system level backups are used, because the user is responsible for managing the recovery of NOT LOGGED objects. It is possible to create an image copy of a NOT LOGGED object, but if the object is lost due to a media failure, application roll back, or abend, any updates applied to the object since the image copy was created are lost. The user is responsible for applying recent updates manually or rebuilding from scratch.

If the object was set to NOT LOGGED when the BACKUP SYSTEM utility was run, RESTORE SYSTEM leaves the object in Recover Pending or Rebuild Pending state. Thus, after running RESTORE SYSTEM, the NOT LOGGED objects must be recreated, rebuilt, or recovered from an image copy. In other words, storing NOT LOGGED objects in the DB2 copy pool is a bad idea because you cannot recover NOT LOGGED objects from the system level backup, and it wastes space in the copy pool back up. If you store the NOT LOGGED objects outside of the DB2 copy pool, RESTORE SYSTEM leaves the NOT LOGGED objects alone. Then if the NOT LOGGED objects were unaffected by a copy pool hardware failure, the NOT LOGGED objects does not require any recovery action.

Therefore, if you intend to leave an object permanently as NOT LOGGED, do not store the object in the DB2 copy pool, because it wastes backup space.

If a NOT LOGGED object is stored in the DB2 copy pool, it is possible with DB2 9 to use the RECOVER utility to exclude the NOT LOGGED objects from the recovery procedure. Because LISTDEF does not have a special keyword for NOT LOGGED, it may be helpful to store the NOT LOGGED objects in a separate database. There is no reason to exclude the NOT LOGGED objects if there is a desire to recover both LOGGED and NOT LOGGED objects, however, because the RECOVER utility automatically distinguishes NOT LOGGED objects from LOGGED objects, and it chooses the system level backup accordingly to restore the objects.

Indexes

The COPY parameter of CREATE INDEX indicates whether the COPY utility is allowed for the index. COPY YES allows full image, concurrent copies, or the use of the RECOVER utility on the index. COPY NO is the default.

SYSLGRNX

SYSLGRNX entries are created whenever an object is opened for update. These entries are updated when the object is closed. The Recover utility uses these entries to determine which log intervals need to be read.

For page sets defined with CLOSE=YES, the PCLOSEN or PCLOSET DSNZPARM parameters control how frequently such page sets are closed. PCLOSEN indicates the number of consecutive DB2 checkpoints since a page set or partition was last updated, after which DB2 converts the page set or partition from read-write to read-only. PCLOSET is the number of minutes since a page set or partition was last updated, after which DB2 converts the page set or partition from read-write to read-only. If the condition for PCLOSEN or PCLOSET is met, the page set or partition is converted from read-write to read-only, and SYSLGRNX is updated accordingly.

Maintaining a page set or partition as read-only is better for the performance of data sharing systems, as well as for recovery, but there is also a cost to switch a page set from read-write to read-only, especially in a data sharing system. That cost is in terms of CPU time, but there is also a cost in terms of the number of entries in SYSLGRNX. As SYSLGRNX grows, the time for Recover to read it grows.

Therefore, to help speed up of Recover and Modify Recovery, SYSLGRNX should be reorganized at least twice a year. Also, run Modify Recovery frequently to remove unneeded entries in SYSCOPY.

The default for PCLOSEN is 5 checkpoints and the default for PCLOSET is 10 minutes. These values are too low for most data sharing systems.

Therefore, assuming the checkpoint frequency is 3 minutes, it is suggested to increase PCLOSEN to 15 system checkpoints and 45 minutes if you are using data sharing.

Fast Log Apply

Fast Log Apply is something that should always be used for recovery and, beginning with DB2 V8, online REORG. The LOGAPSTG DSNZPARM represents the maximum amount of DBM1 storage that can be used by Fast Log Apply. The maximum possible value for LOGAPSTG is 100 MB and that is the default with DB2 Version 8. If set to a lower number, consider increasing to 100 when not short on real storage.

Each RECOVER or REORG utility job that uses Fast Log Apply allocates a 10 MB buffer from the LOGAPSTG pool. Thus, if you specify LOGAPSTG=100 MB, ten jobs can use Fast Log Apply. If any other job needs to apply log records and no Fast Log Apply storage is available, it falls back to normal log apply processing. Each DB2 member has its own Log Apply storage. Thus, if each member uses LOGAPSTG=100 MB, they can each have ten jobs using Fast Log Apply.

The RESTORE SYSTEM utility is unique in that it allocates 500 MB of FLA buffers, independent of the LOGAPSTG parameter. It can use that much storage because RESTORE SYSTEM serializes the entire DB2 subsystem and its data sharing group.

Fast Log Apply has three advantages over normal the log apply process:

- I/O across multiple objects is overlapped, and the log I/O is also overlapped with database I/O.
- Because the pages are sorted within an object, the seek distances on the disk are minimized, reducing the total I/O time

- There is less CPU time because there are fewer I/Os and fewer Getpages.

Fast Log Apply does not support prefetch for indexes.

For best Fast Log Apply performance, it is suggested to avoid recovering more than 100 objects on one recover statement, especially if the objects are large. Do not, however, run more than 10 recover jobs on one DB2 member. This suggestion does not apply to objects which had insignificant update activity. For a larger number of objects, consider spreading the jobs across different data sharing members.

More detail information about using multiple RECOVER jobs is discussed in “Recovery strategies” on page 16.

Comparing RECOVER and RESTORE SYSTEM

RESTORE SYSTEM is convenient for recovering the entire system, but it does not offer any tuning flexibility. Whereas RESTORE SYSTEM is a single utility job running under SYSPITR conditional restart, several RECOVER jobs can be run simultaneously or in a staged manner. When recovering hundreds of objects, Fast Log Apply is faster if the objects are divided among multiple Recover jobs. See “Recovery strategies” on page 16. In some cases, distributing the jobs among different data sharing members helps the performance, because each member has its own Fast Log Apply buffers and its own CPU bandwidth.

It is suggested to use the RECOVER utility instead of the RESTORE SYSTEM utility in order to optimize and tailor the recovery process to meet your specific performance and availability needs.

Be aware of the limitations of RESTORE SYSTEM when it comes to NOT LOGGED objects that were described at “NOT LOGGED” on page 14.

Recovering the DB2 catalog and directory

When recovering the DB2 catalog, ensure that the size of BP0 is at least 10 000 buffers. The default for BP0 in DB2 V8 and DB2 9 is 20 000 buffers.

Recovery strategies

Backup and recovery strategies go hand in hand, but because the evaluation of a backup strategy is dependent on knowing how long recovery takes, it is a good idea to begin with an analysis of recovery performance. The focus here is on recovering the entire system to the current point in time.

The moment to determine a recovery strategy is not when a hardware failure has occurred. The recovery strategy and backup strategy need to be prepared at the same time, although some last minute adjustments might be made to the recovery strategy after the failure occurs. As the backup and recovery strategies are being developed, one should estimate how long recovery will take. As the workload and database grow, this estimate needs to be revised and factored into the backup strategy. In effect, the development of a good backup and recovery strategy is a circular feedback mechanism. One may start with a rough idea how the backups will be done based on rules of thumbs and then develop a recovery strategy based on the initial backup strategy. Yet rules of thumb cannot apply to every situation. One must be prepared to alter the backup strategy after determining if the recover strategy meets the recovery performance objectives.

There is no good substitute to testing the recovery procedures, but testing recovery is expensive because it requires duplicating the production system to create a test system. In lieu of testing a full recovery scenario, developing some simple test cases may help to determine the speeds of your hardware. Generally, the recovery phase is not dependent on the data content. A subset database could be created and dumped to tape using the COPY utility, and recovered using the RECOVER utility. Only the number of objects and the size of the objects are relevant data characteristics that affect restore time.

One of the decisions to make is whether to phase the recovery by workload. For example, if one particular workload was more important than others, it may be brought back online more quickly if the initial recovery phase focused on recovering those database objects needed by the most important workload. The recovery of other database objects can be deferred until after the first workload is brought back online.

Given that system-level objects are used, a decision is made whether or not to use RESTORE SYSTEM. Because RESTORE SYSTEM can recover a subset of the databases, it does not allow for a phased recovery. RESTORE SYSTEM also cannot be used to recover NOT LOGGED objects or objects that have been REORGed since the last time that the system level backup was incremented.

The emphasis in this section is on how to make the best use of the RECOVER utility to minimize the recovery time. Nevertheless, the methodology used to predict the recover time and to adjust the frequency of system level backups is largely the same for RESTORE SYSTEM.

We assume that DB2 9 is installed because RECOVER in DB2 V8 cannot recover objects from a system level backup, and BACKUP SYSTEM in DB2 V8 cannot incrementally update a backup.

Using the RECOVER utility, the objects may be divided among multiple jobs. There are a number of reasons for using multiple RECOVER jobs:

- Employ more FLA buffers and more subtasks in order to speed up Fast Log Apply.
- Overcome the CPU bottlenecks of a single LPAR during FLA.
- Enable multi-tasking for the allocation and open/close operations of image copies.

The primary value of employing more FLA buffers is to reduce the number of Getpages and the number of I/Os, which reduces the amount of CPU time. Secondly, by sorting a set of pages by RID, the DASD seek times can be reduced.

Database I/O during Fast Log Apply

The methodology begins by analyzing and predicting the database I/O time for FLA. The prediction of I/O time is independent of the number of RECOVER jobs. The methodology described here applies specifically to random updates or inserts, not to sequential updates or inserts. Sequential updates and inserts are typically CPU bound, not I/O bound.

It is useful to monitor the amount of log data being generated between backup cycles and adjust the backup frequency accordingly. This simple level of analysis can provide a crude upper bound on the estimate of recovery time, and some simple rules of thumb can be used to divide the objects among recover jobs, but one cannot expect these rules of thumb to be applicable. In order to optimize the recovery time and estimate the time more accurately, it is necessary to do some object level analysis. In order to do some object level analysis, it is necessary to generate incremental image copies. The size of the incremental copies is a measure of how many pages are updated during each backup cycle. As long as the update frequencies remain steady, it is not necessary to regularly generate incremental image

copies. In fact, if system level backups are being incremented frequently (for example, once per hour), it is perhaps not worth the effort to generate incremental image copies ever, because one can take it on faith that recover time is short.

The number of pages for each object is an indirect measure of the number of log records associated with each object. The number of log records is an indication of the amount of CPU time and the amount of FLA buffer space used for each object. Because each RECOVER job is allocated the same amount of buffer space, the buffer space is used more effectively if the log records are divided evenly among the jobs. Furthermore, dividing the CPU time equitably among the jobs and dividing the jobs among different LPARs may enable the CPU time to be divided among different LPARs.

If the objects are divided among multiple RECOVER jobs, armed with the knowledge of how many pages were changed in each object, it is possible to divide the objects among different RECOVER jobs in a manner that divides the pages evenly among the jobs. Dividing the objects in this manner may be at odds with the convenience of the way that LISTDEF INCLUDE/EXCLUDE lists are generated, but it is not unusual for there to be a large skew in the way that updated pages are distributed. To take an extreme example, suppose that two tablespaces of equal size have 200 partitions and suppose that two RECOVER jobs are used, but suppose that 90% of the updates are contained in one of the tablespaces. Although it may be convenient to create one RECOVER job for each table space, the recovery time can be reduced by splitting the partitions of each table space equally among the two jobs. In fact, if a RECOVER job were to restore a million objects that had no log records, the FLA buffers would never be used. Given a 90/10 split, there is more optimization for 10% of the work and the other 90% of the work might be poorly optimized. In conclusion, a heavy update skew can negate the value of having divided the objects into multiple RECOVER jobs.

Besides being an indirect measure of the number of log records, the number of pages is also an excellent indicator of buffer hit ratios and DASD cache hit ratios. Assuming that the buffer pools are much smaller than the DASD cache, the buffer pool residency time is much smaller than the cache residency time. If the cache residency time for a page is 10 minutes, and if the same page is updated in less than 10 minutes, the second update is a cache hit. Now let's make the assumption that a page is either hot or cold, meaning that a hot page stays in the cache throughout recovery, and a cold page is one that is never updated a second time. Let's also assume that the total number of changed pages far exceeds the buffer pool size and the cache size, so that the majority of changed pages must be cold pages. (If the number of changed pages were smaller than the cache size, one could then ignore the I/O analysis of FLA. With these assumptions, the number of changed pages is the number of cache misses, and the number of log records does not affect the number of cache misses.

The next question is how the number of cache misses affects the elapsed time of recovery. It is known that the typical disk service today is in the range of 4 ms to 10 ms, depending on seek distance and disk model. Fast Log Apply has the effect of lowering the average seek distance. Recent studies of throughput have been done using disk drives that IBM delivers in the DS8000 spin at 15 000 RPM and have 300 GB capacity, which is perhaps the most common type of disk drive in the marketplace today. These studies show that the maximum throughput for one RAID 5 rank ranges from 1800 to 2500 I/O read operations per second, depending on the average seek distance. (Response time is poor at these rates.) Because RAID 5 updates consume about four times the disk bandwidth, we can conservatively assume that each RAID 5 rank can do 400 to 500 updates per seconds. This conservatively assumes that every read is a cache miss.

You can think of each changed page in the incremental data set as being one cache miss. Armed with the knowledge of how many pages were changed and knowing the number of RAID 5 ranks in the disk subsystem, you can estimate how long it takes for Fast Log Apply to

update the database. If the time does not meet your objectives, then you may need to adjust your backup cycle time accordingly.

It is suggested to perform object level analysis of the size of incremental image copies during each backup cycle in order to predict the recovery time and adjust the backup cycle time accordingly.

Log I/O

Log I/O time can be largely ignored when the entire system is being recovered, especially with DB2 9. Having increased the number of input log buffers from 15 to 120, DB2 9 can read active log data sets at speed up to 160 MBps, more if the active log data sets are striped. Archive log data sets on disk can also be read fast if DFSMS compression is used.

Multiple RECOVER jobs

Previously we did not explicitly adjust the estimate of I/O time based on the effects of multiple RECOVER jobs. Recall that Fast Log Apply sorts all of the RIDs for each object that is contained in a 5 MB buffer and uses List Prefetch to read the pages. When a large list of objects is divided among, for example, 10 jobs, each job has its own 5 MB buffer and the RIDs become more sorted. The effect of improved sorting may reduce the average I/O service time per page, but this effect is difficult to predict. In the methodology described above, 33 ms of service time was assumed. You have to use your own judgment and knowledge of your hardware to adjust this estimate, and you also have to use judgment as to the effect of sorting the RIDs on the database I/O time.

Nevertheless, the value of sorting the RIDs is much more than I/O time reduction. Another major effect of sorting the RIDs is a reduction in CPU time, because the number of GETPAGEs and the number of I/Os are reduced. To the extent that using multiple RECOVER jobs reduces the number of GETPAGEs and the number of I/Os, it reduces the CPU time.

For example, suppose that a table space is among 100 objects processed by one RECOVER job. Suppose that out of each 5 MB buffer, this table space has 160 RIDs which collapse down to 80 GETPAGEs. Because FLA can read up to 64 pages in one I/O, there are two I/Os for this table space. Now suppose that these 100 objects are split into two jobs. Now there might be 320 RIDs for this table space in each 5 MB buffer and they might collapse to 100 GETPAGEs and all 100 page can be prefetched in two I/Os. Compared to using one job, this represents a 1.6 to 1 reduction in the number of GETPAGEs and a 2 to 1 reduction in the number of I/O operations.

It is unrealistic to try to predict how the effect of multiple jobs will actually reduce the CPU time. However, users should concern themselves with whether or not they have sufficient CPU capacity. Users should also be aware of the number of threads that each RECOVER job might use. For one DB2 subsystem it is a good idea to spread the objects among 10 RECOVER jobs in one DB2 member.

It is suggested to spread all of the objects among 10 RECOVER jobs. Never go beyond 10 RECOVER jobs unless you have a data sharing group and can divide the jobs among different DB2 members.

When deciding how to distribute the objects among different RECOVER jobs, try to spread the large objects among the jobs, especially those large objects that had a lot of updates, so that the FLA storage is used efficiently.

Restoring the database

The advantage of FlashCopy for RESTORE SYSTEM is that the restore phase can complete very quickly and the Fast Log Apply phase can begin as soon as the FlashCopy restore sessions are established. This may take several seconds or minutes given lots of volumes, but this time is not significant. The volumes are physically copied in the background while and after Fast Log Apply is executing. Thus, FLA may take longer than it ordinarily would. No studies have been made to measure and compare the performance of FLA during RESTORE SYSTEM versus a large number of RECOVER jobs.

Let's now discuss the restore phase of RECOVER. RECOVER works somewhat differently depending on whether the full image copies are stored on tape or DASD, or if a system level backup is used. In all cases here let's assume that PARALLEL is specified on the RECOVER statement and also assume that the incremental image copies are stored in disk.

It is complicated to predict the aggregate bandwidth of a DS8000 subsystem. Each DS8000 has three components that contribute to the aggregate bandwidth:

- RIOS bus(es)
- device adapters (DAs)
- channel paths

Any one of these components can limit the bandwidth. So, in order to determine the bandwidth of a DS8000 subsystem for Recover operations, we have to consider the bandwidth of each component.

One of the biggest performance differences between the DS8100 and the DS8000 pertains to the RIOS bus, because each LPAR in the DS8300 has its own RIOS bus, giving the DS8300 twice the bus bandwidth of a DS8100. As Table 1 shows, the bus bandwidth for the DS8100 is 2000 MBps and the bus bandwidth for the DS8300 is 4000 MBps. However, writes consume roughly double the bus bandwidth of reads, because the DS8000 writes the data in two places in cache. Therefore, when restoring from tape to the DS8100 or DS8300, the effective bandwidth of the bus is 1000 or 2000 MBps respectively. When restoring from the DS8100 or DS8300 to itself, the effective rate that data can be copied is about 666 or 1333 MBps respectively.

Table 1 Throughput rates in MBps f or DB2 data

	READ only	WRITE only	COPY
3592 bus	450	450	N/A
DS8000 channel path	275	138	206
DS8000 DA	356	356	138
DS8100 RIOS bus	2000	1000	666
DS8300 RIOS bus	4000	2000	1333

Device adapter and channel path considerations are the same for the DS8100 and DS8300. The number of device adapters is a function of the number of disks; the DS8000 comes with one DA pair for each set of 64 disks, up to a maximum of eight DA pairs. For example, if a DS8000 has 128 disks, then it has two DA pairs, or four DAs. Table 1 shows that the bandwidth for one DA is about 356 MBps. For example, the bandwidth of four DAs is then 1425 MBps. When doing parallel restores from tape to a DS8000 with four DAs, the aggregate DA bandwidth exceeds the bus bandwidth of a DS8100 but not the DS8300. When doing parallel restores from itself to the DS8000, the DAs can copy 712 MBps, which is again

intermediate between the DS8100 and DS8300 bus bandwidth. So, given a DS8100 with four DAs the DAs would not limit the throughput of restores, and given a DS8300 with four DAs the bus would not limit the throughput of restores.

An LCU can have at most eight channel paths. Here we assume that every LCU in the same DS8000 shares the same paths. It is possible that two LCUs in different DS8000s could also share the same channels, but for simplicity we assume that the channels are not shared. Using FICON Express 4 channels, each DS8000 channel path can read up to 275 MBps and write up to 138 MBps. Thus, when restoring from tape to a DS8000 with eight channel paths, the DS8000 can restore about 1100 MBps. When restoring from a DS8000 to itself with eight channel paths, the DS8000 can still restore about 1100 MBps, because the FICON reads do not impact the FICON writes.

Now, putting this altogether, suppose that a DS8100 has four DAs and eight channel paths. What is the restore rate if the image copies are stored on the DS8100? The bus can restore 666 MBps, the DAs can restore 712 MBps, and the channel paths can restore 1100 MBps. Therefore, in this example, the DS8100 can restore 666 MBps, limited by the RIOS bus.

Next, let's consider restores from one or more 3592 tape subsystems to the same DS8100 as above. By offloading the read operations to tape, this bus bandwidth of the DS8100 can be focused on the writes. Because the bus bandwidth is 1000 MBps, the four DAs become the bottleneck of the DS8100 at 712 MBps. However, because a single 3592 subsystem can only read 450 MBps, two 3592 subsystems are needed to push the restore rate beyond 450 MBps.

If you know the size of your database, you can use your estimate for the restore rate to predict the restore time. For example, if the restore rate is 1000 MBps and you have to restore 1 TB, the restore time is 1000 seconds, or 16.7 minutes. The restore time is proportional to the size of the database.

Because the DS8300 has double the bandwidth of a DS8100, the DS8300 tends to be a better choice for the optimization of recovery performance, especially when there are more than four device adapters and more than eight channel paths. The DS8300 also has double the CPU capacity of a DS8100, which is important for advanced copy services such as FlashCopy and Metro or Global Mirroring.

If there are a lot of small objects, then the time to allocate and open the objects may outweigh the time to copy the data. The allocation and open and close for image copies and incremental copies are serialized by the RECOVER job's control task. One can assume that the RECOVER control task can process about five objects per second. Thus, given 1000 data sets (including all incremental copies), it would take 200 seconds for one RECOVER job to open and close them. This is one of the reasons why some object level analysis helps determine the number of RECOVER jobs to use. If there are 10 000 data sets accessed, it takes approximately 2000 seconds to process all of them, but dividing the objects among 10 RECOVER jobs would reduce the time to 200 seconds.

The time to mount the tapes can be ignored as long as the image copies are efficiently stored on large capacity tapes and also read efficiently. The time to allocate and open the image copy data sets can also be largely ignored provided that there are several RECOVER jobs. For example, suppose that there are a total of 50 000 DB2 objects. Assuming five data sets per second, the estimate for the total open/close time is 10 000 seconds. If 10 RECOVER jobs are used, the estimated open/close time is 1000 seconds per job, or 16.7 minutes. Because this time is overlapped with and less than the data transfer time, the allocation and open time can be ignored if the 50 000 objects contain a few terabytes.

Of course, the fastest method for restoring the database is to use a system level backup. Using the RECOVER utility to restore objects from a system level backup, we can assume the same cost as open and close (it is actually smaller because the HSM requests within a single

RECOVER job are done in parallel). The time for the restore phase of RESTORE SYSTEM is a function of the number of volumes in the DB2 copy pool, but it can largely be ignored.

Data sharing considerations

Each DB2 member in a data sharing group may handle up to 10 RECOVER jobs without sacrificing FLA. What is the optimal number of jobs to use when there are lots of data sharing members? No studies have been done to examine this question, but presumably, if there were 100 jobs each reading 10 different logs and merging them, there would be excessive log I/O and there would probably not be much value. However, if the updates used system affinity (which is unlikely), it is possible that the RECOVER objects could also be organized according to system affinity. Short of such system affinity, the DBAs have to use their own judgment about the total number of RECOVER jobs to use. Object level analysis would help, because out of 10 000 objects, the majority is not likely to affect the FLA buffer consumption significantly. Objects that were not updated should not be counted in the analysis of FLA, although these objects may affect the restore time if there are a lot of them.

Given a large data sharing system and a system level backup, RESTORE SYSTEM is usually the best option to restore the entire system quickly. There are some advantages to using RECOVER, however. One of the advantages of RESTORE SYSTEM is that establishing a few hundred volume level FlashCopy restore sessions is much faster than establishing tens of thousands of data set level FlashCopy restore sessions. In other words, the restore phase can be completed sooner. Because RESTORE SYSTEM also uses five times as much FLA storage as one DB2 member, it would take five data sharing members and 50 RECOVER jobs to equal the FLA storage used by one RESTORE SYSTEM job, and RESTORE SYSTEM only has to read the logs one time. Thus, RESTORE SYSTEM has the potential to achieve better performance than lots of RECOVER jobs, provided that one LPAR has the CPU capacity to effectively execute 1000 tasks. However, following the execution of RESTORE SYSTEM, RECOVER has to be executed for NO LOG objects and objects that were REORGed after the last time that the system level backup was incremented.

Backup strategies

In this section, after introducing some failure scenarios, we define the related recovery strategies.

A hypothetical workload and failure scenarios

In this part we present a hypothetical hardware environment with a hypothetical DB2 database and a hypothetical workload. Then we posit three different failure scenarios. One of these scenarios is a hardware failure and the other two scenarios involve programming errors. In the first two scenarios there is a need to recover to the current point in time and in the second scenario there is a need to recover to a prior point in time.

Figure 5 summarizes the hardware environment and the database characteristics. The environment consists of a single DS8300 subsystem which contains 512 disk drives of 300 GB each and the storage is partitioned into 4096 volumes. Because RAID 5 is the chosen RAID design, the total addressable capacity is 113 TB, which comes out to be about 27 GB per volume. In terms of DB2 database capacity, the addressable capacity is actually 99 TB due to the fact that DB2 can only store 48 KB per track. There are 50 databases defined amounting to a total of 30 TB and there are 50 000 DB2 objects. Each database corresponds to a particular workload. If any part of a database is damaged, the corresponding workload is

inoperable. One of the databases that is the focus of attention is DBXX1. DBXX1 consists of 1 TB.

```
1   DS8300 storage server
512 disk drives, 300 GB each
99 TB of DB2 database capacity
30 TB DB2 database defined
50,000 DB2 objects
4096 volumes
```

Figure 5 Hardware/workload environment: Disk/volume/database configuration

Figure 6 summarizes the tapes characteristics. The environment also has four 3592 tape subsystems each with four tape drives and four FICON Express 4 channels. Tapes may be used for either volume dumps or to hold full image copies. The max capacity of each tape is currently 1 TB. Thus, with careful tape stacking, it takes a minimum of 30 tapes to hold the image copies of all 50 databases. However, because it is decided to split different databases on different tapes, it turns out that 120 tapes are needed. Three tapes are dedicated to DBXX1. The image copies are stored on tape using 256 KB block sizes, which is supported by DB2 9.

```
4 3592 tape subsystems
4 tape drives per subsystem
4 FICON Express 4 channels per subsystem
1 TB/tape capacity
120 tapes hold one set of DB2 full image copies
```

Figure 6 Hardware/workload environment: Tape configuration

Because the DS8300 subsystem has 512 disks, it is fully populated with 16 device adapters. Therefore, the throughput capacity of the device adapters is estimated to about 5.6 GBps and the throughput capacity of the RIOS bus is estimated to be 4.0 GBps. However, because the DS3800 is configured with only four channels, the channels limits the throughput. We estimate that these four channels can write 4×138=552 MBps. Therefore, the estimated time to restore 30 TB of data from tape, or from image copies, is about 15 hours. Doubling the number of channels could cut this time by half. Increasing the number of channels to 16 could reduce the time to less than four hours. The RESTORE SYSTEM utility could do better, because the channels are not used to restore the database. In any case, the restores are done in the background.

Next, let's turn to the question of Fast Log Apply time. To simplify the example, let's assume that the workload is in a steady state, modifying the same amount of data throughout the day.

Figure 7 illustrates what information is needed to predict the Fast Log Apply time and adjust the backup frequencies.

```
372 GB of primary active log space
36 GB/hour
10 hours to wrap the active log space
10 million total modified pages/hour
1 million DBXX1 modified pages/hour
```

Figure 7 Workload update characteristics

One of the key pieces of information is the fact that 10 million pages are modified each hour. This could be determined by generating incremental image copies. Does that mean that incremental copies need to be generated every hour or every day? No. The incremental copies only need to be generated once to develop an estimate of the page modify rate, or as often as desired in order to monitor the changes in the workload. For this example, the incremental copies were generated four hours after the full image copies and it was found that the incremental copy data sets consumed 160 GB. That is, 40 GB per hour, which is 10 million 4 KB pages. (The number of pages could be adjusted for different page sizes.)

The data also shows the correlation between the amount of logging and the number of modified pages. In this example, for every 1 GB of log data, 280 thousand database pages were modified (that is, 10 million pages/hour divided by 36 GB/hour). Because it is much easier to monitor the log activity than the page update activity, one can monitor the log activity to judge how the workload characteristics are changing over time.

Knowing that the number of modified pages per hour is 10 million, it is possible to estimate the amount time to apply the log records based on the throughput capacity of each RAID rank. A RAID 5 rank consisting of 8 disks of the type illustrated here (300 GB, 15 000 RPM) can do about 1500 to 3000 read operations per second, depending on the seek distance. Conservatively assuming 1600 IO per second, and then allowing time for the data and RAID 5 parity to be updated, we can divide the read throughput by four, resulting in 400 data pages updated per second per rank. With 512 disks divided among 64 ranks, the aggregate update rate is 25600 updates per second, provided that the updates are evenly distributed among the ranks. With these assumptions, it takes 6.5 minutes to update 10 million pages (an hour's worth of workload updates). You could refine this estimate if you analyze how the updates are distributed among the ranks.

We can also estimate the amount of time for Fast Log Apply to read the DB2 log. Because 372 GB of log data is written every 10 hours, 37.2 GB are written each hour. DB2 can read the log at about 130 MBps. Thus, it takes about 4.8 minutes to read an hour's worth of log records. Because the log time is less than the database I/O time and because the log I/O and database I/O are overlapped, the log I/O time can be ignored. This is typically true unless the database is spread across a vast number of disks.

We have assumed thus far that no archive logs need to be read during recovery. We want to choose a backup methodology that avoids having to use archive logs. There are 372 GB of active dual log space (that is, 372 GB for primary and 372 GB for secondary), which is the maximum that DB2 allows. The workload runs 24×7. The workload generates 36 GB of log space every hour. Therefore, the active logs hold about 10 hours worth of log records, DB2 writes the archive logs to disk, but HSM migrates the archive logs to tape daily. The log records are divided among all of the DB2 objects.

Now let's consider a few failure scenarios in the context of this environment.

- Media failure

  At 10 p.m. Thursday, a disk drive failed, and though RAID 5 protected us against a single disk drive failure, a second drive in the same RAID rank failed before we could replace the first one. New disk drives are quickly installed. We do not know which objects were allocated on those disk drives. It may be possible to determine which objects were lost and to recover just the lost objects, but it is considered faster to simply recover all of the tablespaces.

- Recover to current

  At 10 p.m. Thursday, a volume was accidentally reformatted and that volume contained part of the production database. Because it is unknown which data sets were on that volume, every table space needs to be recovered.

- Recover to prior

  At 10 p.m. Wednesday, a change was made to one of the application programs. In some cases, it mistakenly began updating the wrong rows in DBXX1. No other database was affected. Unfortunately, the mistake was not detected until Thursday afternoon. The application is quickly fixed, but now DBXX1 needs to be fixed. DBXX1 needs to be recovered to 10 p.m. Wednesday, and then the application can be rerun.

The first scenario is the one which most people think about when they are planning their backup strategies. The second scenario is atypical, but it is described here to just illustrate a point. The third scenario is a common (usually due to lack of testing at cycle time) example illustrating what must be done to recover some of the database to a prior point in time.

Some backup strategies

Next, we illustrate five different backup strategies. For each of these strategies, we estimate the restore time and the Fast Log Apply time. Armed with that information, we can later posit some recover performance capabilities for the hardware and calculate the time to recover. These different strategies also affect the performance and storage capacity for the online work.

- Strategy 1: NEWCOPYNO

  Write full image copies each Sunday at 1 a.m. to tape. Each day of the week at 12 p.m., 4 p.m., and 8 p.m., incremental copies are created. At 1 a.m. each day, MERGECOPY is run with the NEWCOPY NO option, and then another set of incremental copies are created. Thus, there are 4 sets of incremental copies created each day.

- Backup strategy 2:NEWCOPY YES

  The second strategy is the same as the first one except that instead of NEWCOPY NO, we use NEWCOPY YES which creates full image copies, and no extra incremental copy is needed at 1 a.m. The full image copies are written to tape. There are still 4 sets of incrementals for each day.

- Backup strategy 3: COPY FULL

  The third strategy is the same as the previous two, except that instead of using MERGECOPY, COPY FULL is run every night at 1 a.m. There are 3 sets of incrementals for each day.

- Backup strategy 4: Mix system backup and database

  In the fourth strategy, we divide the volumes in half and use the BACKUP SYSTEM utility to create a system level backup. The backup is incremented once each hour. The backup volumes are spread across all of the RAID ranks. As with the third strategy, full image copies are written to tape each night at 1 a.m., but no object level incremental copies are created.

- Backup strategy 5: Isolate system backup from database

  The fifth strategy is similar to the fourth, except that the backup volumes and database volumes are segregated on different RAID ranks.

Notice that aside from the performance effect of running the COPY utility or BACKUP SYSTEM utility, all of these backup strategies have the same affect on workload performance except for the fifth strategy. In order to segregate the databases and backup volumes, the databases only have access to 256 disks instead of 512 disks. Therefore, disk response time with the fifth strategy tends to be higher.

Recover actions

Table 2 summarizes the actions that need to be taken in order to recover from our three different failure scenarios, given each of the five different backup strategies.

Table 2 Recover actions

	Failure 1: Media failure	**Failure 2: Recover to current**	**Failure 3: Recover to prior**
Backup strategy 1: NEWCOPY NO	**Action:** RECOVER all databases **Restore:** 30 TB from 120 tapes +4 increments **Log Apply:** 20M pages	**Action:** RECOVER all databases **Restore:** 30 TB from 120 tapes +4 increments **Log Apply:** 20M pages	**Action:** RECOVER DBXX1 **Restore:** 1 TB from 3 tapes + 4 increments **Log Apply:** 2M pages
Backup strategy 2: NEWCOPY YES	**Action:** RECOVER all databases **Restore:** 30 TB from 120 tapes +3 increments **Log Apply:** 20M pages	**Action:** RECOVER all databases **Restore:** 30 TB from 120 tapes +3 increments **Log Apply:** 20M pages	**Action:** RECOVER DBXX1 **Restore:** 1 TB from 3 tapes + 3 increments **Log Apply:** 2M pages
Backup strategy 3: COPY FULL	**Action:** RECOVER all databases **Restore:** 30 TB from 120 tapes +3 increments **Log Apply:** 20M pages	**Action:** RECOVER all databases **Restore:** 30 TB from 120 tapes +3 increments **Log Apply:** 20M pages	**Action:** RECOVER DBXX1 **Restore:** 1 TB from 3 tapes + 3 increments **Log Apply:** 2M pages
Backup strategy 4: Mix system backup and database	**Action:** RECOVER all databases **Restore:** 30 TB from 120 tapes **Log Apply:** 20M pages	**Action:** RESTORE SYSTEM **Restore:** 30 TB from disk **Log Apply:** 10M pages	**Action:** RECOVER DBXX1 **Restore:** 1 TB from 3 tapes **Log Apply:** 21M pages
Backup strategy 5: Isolate system backup from database	**Action:** RESTORE SYSTEM **Restore:** 30 TB from disk **Log Apply:** 10M pages	**Action:** RESTORE SYSTEM **Restore:** 30 TB from disk **Log Apply:** 10M pages	**Action:** RECOVER DBXX1 **Restore:** 1 TB from 3 tapes **Log Apply:** 21M pages

In the case of a media failure, we could spend some time trying to analyze which specific objects resided on the failed media, but a tool would be needed to perform such analysis quickly. For example, we could dump the VTOCs nightly and match those VTOC listings with the DB2 catalog, but that approach would not identify objects that had been created recently. A safer approach is to use the DB2 catalog to generate a full list of objects to be recovered, then scan the VTOCs of every volume that was not lost. The objects found to be intact could be deleted from the object list. The only problem with this approach is that one has to be

careful about multi-volume data sets. If an object was a multi-volume data set and any one of the volumes was on the failed media, then the entire object must be recovered. To deal with multi-volume data sets, the system ICF catalogs must be analyzed rather than the VTOCs. For each data set in an ICF catalog, the list of volumes must be scanned to see if any one of them is in the list of volumes that were lost. In the final analysis, it is probably a lot simpler, though often longer, to just recover every table space.

Table 3 summarizes the estimates of the recovery time, including the time for both the restore phase and Fast Log Apply phase, for each combination of failure scenarios and backup strategies.

Table 3 Recovery estimates

Backup strategy		Failure 1: Media failure	Failure 2: Recover to current	Failure 3: Recover to prior
1	Restore	333	333	11.1
	Fast Log Apply	43	43	4.2
	Total	376	376	13.2
2	Restore	333	333	11.1
	Fast Log Apply	43	43	4.2
	Total	376	376	13.2
3	Restore	333	333	11.1
	Fast Log Apply	43	43	4.2
	Total	376	376	13.2
4	Restore	333	0	11.1
	Fast Log Apply	43	10.8	22.5
	Total	376	10.8	22.5
5	Restore	0	0	11.1
	Fast Log Apply	21.5	21.5	45.0
	Total	21.5	21.5	46.1

The fourth backup strategy cannot use the RESTORE SYSTEM utility to recover from the media failure because part of the system backup was lost. The only backup strategy which is capable of handling the media failure without having to restore from tape is the fifth strategy. Also, because incrementing the system level backup is less obtrusive than using the COPY utility to create incremental copies, we were able to refresh the system level backup hourly, and therefore when we do need to run the RESTORE SYSTEM utility, the amount of log records that need to be applied is greatly reduced. On the other hand, because the fifth backup strategy constrained the databases onto fewer disks, transaction performance and query performance was compromised.

The actions needed to recover from the accidental reformatting of one volume is much the same as the actions that are needed to recover from a media failure, but for one exception. The fourth backup strategy can take advantage of the RESTORE SYSTEM utility, because the backup is completely intact.

The third failure scenario is the most difficult one from which to recover, even though DBXX1 is the only database that is affected. Two factors make it difficult:

- We have to recall the archive log data sets from tape to apply the log records from the previous night. The recall times are not shown in Table 3 on page 27.
- The system level backup cannot be used to restore DBXX1, because the error occurred prior to the system backup being incremented.

Had the fourth and fifth backup strategies continued to incorporate object-level incremental copies, the log apply time would be have been the same as the first three strategies.

Overlooked in the above analysis is the fact that after running the RESTORE SYSTEM utility, any object that had been reorganized after the system level backup was created needs to be recovered using the RECOVER utility. After having reorganized a lot of objects, it would be best to increment the system level backup quickly in order to minimize the likelihood of having to recover from image copies.

If the objective was to be able to recover the entire system in one hour after a media failure, the only strategy that could meet this goal is the last one, namely to use a system level backup.

Summary

This paper provides a broad set of recommendations for creating backups and for recovering from different failure scenarios. Some decisions involve trade-offs that affect transaction performance as well as the recovery time for different failure scenarios. The decisions begin on the day that the storage is purchased and configured. Such decisions also affect the probability of ever having to recover from a backup.

Both object level backups and system level backups were considered, either as distinct solutions or in combination. Both full and incremental backups were considered. The advantages and disadvantages of using tape were considered, both from a cost perspective and a performance perspective. Given that tapes are used to store the backups, the pros and cons of different methods of tape management were discussed.

Recovering from backups is something nobody wants to have to do, but when it happens, there are some decisions to be made on how best to recover. Those decisions should be discussed and planned before the failure occurs. If the RECOVER utility is going to be used, the recovery of objects may be staged, or recovered in parallel. A methodology for predicting the recovery time was presented that can be used to adjust the backup frequency. Nevertheless, even with the accurate prediction techniques, there is no substitute for practice, although practicing recovery is itself an expensive procedure. One should develop the recovery scenarios and determine the acceptable recovery time. Practice helps determine if procedures are in place to meet the recovery requirements of each failure scenario.

The team that wrote this Redpaper

This paper was produced by a team of specialists working at the Silicon Valley Lab, San Jose California.

Jeff Berger is a member of the DB2 for z/OS performance department in IBM Silicon Valley Laboratory. For most of his 29 years in IBM, Jeff has worked on system performance of IBM mainframes, specializing in DASD storage and database systems, both hardware and

software. He has written two other Redpapers: *How does the MIDAW Facility Improve the Performance of FICON Channels Using DB2 and other workloads?* REDP-4201, and *Index Compression with DB2 9 for z/OS*, REDP-4345. Jeff has contributed several patents and several papers, which were published by Computer Measurement Group, DB2 IDUG Solutions Journal, and IBM z/OS Hot Topics.

Paolo Bruni is a DB2 Information Management Project Leader at the International Technical Support Organization based in Silicon Valley Lab. He has authored several Redbooks about DB2 for z/OS and related tools, and has conducted workshops and seminars worldwide.

Many thanks are owed to the following people for their thoughtful review of this paper:

John Campbell
Bill Franklin
Laura Kunioka-Weis
Tom Majithia
Haakon Roberts
Judy Ruby-Brown
Johannes Schuetzner
Akira Shibamiya
Bryan Smith
Silicon Valley Lab, San Jose CA

Lee La Frese
Glenn Wilcock
Storage Software, Tucson AZ

References

- *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370

  by Paolo Bruni, Pierluigi Buratti, Florence Dubois, Judy Ruby-Brown, Christian Skalberg, Tsumugi Taira, Kyungsoon Um
- *DB2 9 for z/OS Technical Overview*, SG24-7330

  by Paolo Bruni, Roy Cornford, Rafael Garcia, Sabine Kaschta, Ravi Kumar
- *DB2 9 for z/OS Performance Topics*, SG24-7473

  by Paolo Bruni, Kevin Harrison, Garth Oldham, Leif Pedersen, Giuseppe Tino
- *IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786-03

  by Bertrand Dufrasne, Wilhelm Gardt, Jana Jamsek, Petre Kimmel, Jukka Myyrylainen, Markus Oscheka, Gerhard Pieper, Stephen West, Axel Westphal, Roland Wolf
- *IBM System Storage DS8000 Series: Copy Services with IBM System z*, SG24-6787-03

  by Bertrand Dufrasne, Wilhelm Gardt, Jana Jamsek, Peter Kimmel, Jukka Myyrylainen, Markus Oscheka, Gerhard Pieper, Stephen West, Axel Westphal, Roland Wolf
- *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation,* SG24-6789-02

  by Babette Haeusser, Soren Aakjar, Jim Fisher, Mauro Silvestri
- *DFSMShsm Fast Replication Technical Guide*. SG24-7069

  by Mary Lovelace, Martin Berger, Enete G. S. Filho, Bengt Larsson
- *DB2 Version 9.1 for z/OS Utility Guide and Reference,* SC18-9855

Archived

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright International Business Machines Corporation 2008. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This document REDP-4452-00 was created or updated on October 10, 2008.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DB2®
DFSMS™
DFSMSdss™
DFSMShsm™
DS8000™
ESCON®
FICON®
FlashCopy®
IBM®
Redbooks®
Redbooks (logo) ®
System Storage™
System z®
TotalStorage®
z/OS®

Other company, product, or service names may be trademarks or service marks of others.

Archived