IBM® Smart Analytics System

Best Practices

Using IBM InfoSphere Optim High Performance Unload as part of a Recovery Strategy

Garrett Fitzsimons
IBM Data Warehouse Best Practices Specialist

Konrad Emanowicz
IBM DB2 Data Warehouse QA Specialist

Richard Lubell
IBM Information Development Specialist

Issued November 2011

Executive Summary

This paper is targeted at those involved in planning, configuring, designing, implementing, or administering a data warehouse based on DB2® Database for Linux®, UNIX, and Windows® software. The examples in this paper apply generally but are focused on the IBM ®Smart Analytics System based on System x® and Power Systems™ servers.

IBM InfoSphere Optim® High Performance Unload (HPU) for DB2 for Linux, UNIX, and Windows V4.2 is a high-speed tool for unloading, extracting, and repartitioning data in DB2 for Linux, UNIX, and Windows databases. HPU is designed to extract data from DB2 incremental and delta backup images and from DB2 table space containers. HPU has additional uses in data migration and repartitioning that will be dealt with in future paper.

The focus of a database backup and recovery strategy should be the recovery plan. The paper *Building a Recovery Strategy for an IBM Smart Analytics System* discusses best practice recommendations for backup and recovery using the native DB2 software commands. Although not recommended for full database recovery, HPU can be effective in speedily recovering discrete pieces of data from tables and table spaces.

The database is not affected by the HPU process and table spaces do not need to be placed in an offline state. HPU can therefore offer advantages over other recovery methods in data availability and processing speed. Incorporating HPU into your recovery strategy for scenarios involving data loss or corruption can help achieve service level objectives for data availability and data recovery.

Using the recommendations in this paper can help you develop a strategy that uses HPU to complement your existing recovery strategy.

Introduction

This paper describes best practices for incorporating the use of HPU into your recovery strategy and implementing HPU for an IBM Smart Analytics System data warehouse. The paper covers how to unload data from backup images.

To use this paper, you should have a basic knowledge of HPU software as well as DB2 software as implemented in an IBM Smart Analytics System. The further reading section in this paper contains links to product documentation and papers referenced throughout the paper.

The first section of the paper discusses how to incorporate HPU into a recovery strategy and when to use HPU to meet your recovery objectives.

The second section contrasts the use of output files and named pipes and discusses introducing parallelism.

The third section reviews the HPU command and control file and recommends best practices for creating control files.

The fourth section of the paper details how to install and configure HPU in an IBM Smart Analytics System.

The appendixes outline the configuration of the test system used in developing this paper and also contain examples of using HPU in common recovery scenarios.

This paper builds on and complements a series of best practices papers that discuss aspects of a DB2 data warehouse. Refer to these papers for further information.

Incorporating HPU into your recovery strategy

HPU is a high-speed stand-alone utility for unloading, extracting, and repartitioning data in DB2 for Linux, UNIX, and Windows databases. HPU unloads large volumes of data quickly by reading directly from full, incremental, and delta backup images or table space containers rather than through the DB2 software layer. HPU does not replace other recovery tools in your recovery strategy, but can have significant advantages in recovering discrete volumes of data.

Incorporate HPU into your recovery strategy to provide:

- Reduced recovery times through targeted data recovery

  Use the WHERE clause when selecting data from backup images to target only the data that was lost, compromised, or corrupted, reducing recovery time.

- Increased data availability and reduced resource usage during recovery scenarios

  Unload data from a backup image and optionally process this data on a non-production system, maintaining availability of the query workload and reducing the effect on the production system.

- Integration with DB2 and Tivoli Storage Manager software

  Query TSM to unload data from full, incremental, and delta backup images archived to TSM by the DB2 `BACKUP` command, reducing errors and facilitating outside recovery.

Install HPU only on the physical servers where you intend to run HPU to unload data.

HPU would be useful, for example, in a recovery scenario where an ETL application change has compromised data such as a customer ZIP code. Instead of using the RESTORE command and affecting data availability, HPU could be used to unload the `customer id` and `zip code` columns from a backup image. The unloaded data would then be loaded into a temporary table and the customers' ZIP code corrected by using a simple `UPDATE` statement.

When to use HPU

HPU can complement your existing recovery strategy where:

- The Recovery Time Objective (RTO) is aggressive.

  HPU does not need to connect through the DB2 software layer but instead reads data directly from the backup image, increasing the speed of unloading data**.** Use HPU to unload data from a backup image to multiple output files in parallel for one or more tables. Use the DB2 Load utility to complete the recovery of the dropped table.

  For example, the method of recovering each dropped table in separate sequential restore operations is not compatible with some recovery time objectives (RTO). HPU can perform data unload from multiple tables in parallel and does not need to perform a **`ROLLFORWARD`** to end of logs, helping to ensure faster data recovery times.

- The database needs to be available to queries during the recovery process.

  HPU unloads data directly from backup images instead of through the database engine. You can thereby recover data for a table without the need to put a table space or database offline.

HPU can unload data for individual table or for all tables. By default, HPU unloads data for all tables in the table space in parallel, reducing the effect on query workload by minimizing network traffic. This approach helps ensure that backup images are read a minimum number of times. HPU uses system resources but will not need to connect to the production database.

Data can be restored by using the LOAD command with an INSERT with SELECT statement. Alternatively, the ALTER TABLE DETACH and ATTACH command can be used to switch the recovered data partition with the corrupted data partition. These operations do not take the system offline as is required when using the Restore command.

- The recovery process needs to take place away from the production system.

  When data needs to be restored and validated outside of the production system, you can unload data for dropped or corrupted tables into a non-production system. This action allows you to cleanse and prepare data for load into the production system. Use this approach to minimize the effect on the production system. HPU supports DB2 software integration with IBM Tivoli Storage Manager (TSM); use this tool to recover and validate data on a non-production system. This topic is discussed further in Appendix B.

  For example, HPU can be used to recover specific pieces of data to a non-production system. When used in this way, HPU must be installed on the non-production system and a copy of the database catalog backup image must be accessible. If only a single column that has been compromised, use HPU to unload just the primary key and the contents of the affected column.

Retain a copy of the most recent database catalog backup image on disk to help increase recovery speed when using HPU.

HPU and backup images

Several factors restrict the volume of data that can be unloaded and the number of backup images that can be accessed in parallel, The factors include the number of tape drives available, the resources assigned, and the backup strategy chosen.

HPU directly reads backup images created by DB2. The following is recommended:

- Use offline backup images where available to help ensure the integrity and consistency of data unloaded.
- Use online backup images only when you are certain that no transactions took place against the objects you are unloading during the most recent online backup.
- Use table space backups instead of full database backups when possible to reduce the size of the backup image you are reading and therefore enable faster data unload times.
- Include the capture of table create statements in your backup strategy. When a table is dropped it needs to be re-created before recovery via a LOAD command.

Use offline backup images or consistent online backup images to help ensure the integrity of the data unloaded.

Your backup strategy should allow for the number of tape drives available per data node. In a partitioned database it is recommended that you back up one database partition per data node in parallel across all data nodes. This recommendation assumes one tape drive per data node. If two tape drives are available, then two database partitions per data node can be backed up in parallel. Through this strategy, a database backup consists of multiple backup images with different timestamps. HPU can use a partial timestamp parameter to reference all of these backup images.

If no backup image timestamp is specified then HPU determines the most recent full backup taken and also identifies incremental and delta backup images created since the full backup. HPU temporarily stages data from the full database backup and then applies any changes to the data found in the incremental or delta backup images before creating the final output file. Note that additional staging space is required when data is unloaded from incremental backup images.

Access to a copy of the database catalog is needed by HPU to determine the characteristics of the tables for which data is being unloaded. To avoid any database connections when using HPU, it is recommended that you retain a backup of the database catalog on disk for use with HPU. HPU can refer to a database that contains the database catalog or to an individual backup image of the database catalog itself.

Using HPU

When using HPU to recover data consider the disk and server resources available and your use of these resources.

Using HPU with output files or named pipes

HPU uses a staging area to stage data pages read from the backup image. Data is then written to output files in the output area.

Unloading data to output files allows this process to be separated from the process of loading data, allowing for validation or postponement of the load operation. Using named pipes allows data to be unloaded from a backup image and loaded to the target database table concurrently and uses less disk space to perform the restore operation.

Choose your recovery method based on your resource priority: use named pipes where there is insufficient disk space on the data nodes, or unload to output files where processor usage is limited.

To facilitate parallel unload and load operations, create a control file to unload data for each database partition to individual output files on the appropriate node. Then use the DB2 Load utility to load data in parallel across all the database partitions, avoiding unnecessary network traffic between data nodes during the recovery process. The syntax or use of the DB2 Load utility is not altered by HPU and all features and parameters operate normally.

When using the DB2 Load utility to complete a recovery scenario it is recommended that you perform a backup of the recovered data in line with your service level objectives for backup and recovery. When using the `LOAD` command it is recommended that you use the `LOAD_ONLY_VERIFY_PART` mode. HPU does not generate partition header data in output files. Use `LOAD_ONLY_VERIFY_PART` rather than the `LOAD_ONLY` mode to indicate to the `LOAD` utility that the data is already partitioned but does not contain a partition header.

Controlling parallelism with the MAXSELECTS parameter

The `MAXSELECTS` parameter determines the number of SELECT statements and therefore the number of tables to be processed in parallel when unloading data from backup images. Using `MAXSELECTS` in this way minimizes the number of times the backup image is read but increases the resources required by the HPU process.

If a value for `MAXSELECTS` is not specified, HPU attempts to process all select statement in the unload block, or all tables in a specified table space.

It is recommended that you include a `MAXSELECTS` entry in the `db2hpu.cfg` configuration file and associate a value of 1. This action prevents an HPU process from consuming more resources than intended. This default value can then be overridden in an individual control file where needed. Use the

MAXSELECTS control file configuration parameter to restrict the number of tables to be unloaded from in parallel where you want to reduce the resources consumed.

Use the configuration and control file parameters to throttle HPU usage of processor resources.

Controlling processor resources with the nbcpu configuration file parameter

The *nbcpu* configuration parameter specifies the maximum number of threads allowed when unloading data. Default behavior is to start a work unit for each processor core. This parameter is set in the `db2hpu.cfg` configuration file and cannot be overridden in the control file.

In a scenario involving a database outage, you might want to allow HPU to use all available processor resources in order to unload data as quickly as possible. In an environment where queries are running during recovery, configure HPU to restrict the resources consumed, reducing the amount of processor used.

How the **db2hpu** command works

The HPU utility uses a command-line interface with a control file to perform the data unload operation.

Data unloads from the backup image to staging files and then writes to output files for use with the DB2 Load utility. Employing a separate staging directory and output file directory is recommended to separate files and input and output operations during unload processing. HPU processes one single database partition per node in sequence on each data node in parallel. All the tables in the table space unload on each data node in the following format: *file.NNN_<tablespace>_<schema>_<table>* where *NNN* is database partition and *<schema>* and *<table>* are the schema and tables names.

Avoid contention when unloading data from tape by ensuring that the sequence of database partitions referenced in the unload blocks mirrors the backup sequence.

These topics are discussed in the following sections with detailed examples in appendix B:

- Using HPU control files
- Creating a control file
- Issuing the **db2hpu** command

Using HPU control files

The structure of the control file consists of two types of control blocks:

- Global Block
- Unload Blocks

Global Block

The global control block contains configuration data that is common to all unload blocks in the HPU control file. There is only one global block per control file and the global block must be the first block in the control file.

The **db2hpu** command references a database catalog to gather details on the tables to be unloaded. Specify the USING BACKUP CATALOG clause when the objects being recovered are not available on the existing database catalog contained in the database referenced. Scenarios for this action include where objects were dropped, the catalog is corrupted or you are executing the **db2hpu** command on a non-production system where DB2 software is not installed.

The following example illustrates the content of a control block where the database catalog backup image for the database TPCDS is to be used. The backup image for the database catalog to be used is located on disk and the backup image to use has all or partial timestamp **2011033108**. A semi-colon is used to terminate the statement and the block.

```
-- Global Block specifies disk location for the database catalog
GLOBAL USING BACKUP CATALOG TPCDS from "/work0/HPU/catalog_BACKUP"
 TAKEN AT 2011033108;
;
```

Unload Blocks

The unload block specifies the table space, tables, and `SELECT` statement for the data to be unloaded. Multiple unload blocks can be used in a single control file to perform a sequence of unload operations. Scenarios for this usage include when unloading from multiple table spaces or from multiple table space backup images.

HPU attempts to read the backup image once per unload block. Multiple `SELECT` statements in a single unload block are processed in parallel subject to the `MAXSELECTS` parameter value.

When creating a control file in an IBM Smart Analytics System:

- Specify the USING BACKUP DATABASE clause as the last one in the UNLOAD TABLESPACE command when unloading the whole table space or the HPU command will fail with incorrect syntax.
- Use a period after the file name in the control file; this notation is required for the DB2 Load utility to load the data successfully.
- Use the OUTFILE clause in preference to the OUTPUT clause.
- Unload data for all tables needed; HPU does not support JOINS.

The following example illustrates how an unload block would be created to unload data from a table space backup image from database partitions 1 and 9 in parallel:

```
-- Unload Block specifies backup image for partitions 1 and 9
-- A table space backup is used.
UNLOAD TABLESPACE
PART(1,9)
USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM ON SOURCE
 HOST TAKEN AT 20110331080240;
-- Select statement specifies customer table
SELECT * FROM BCUAIX.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/bcuaix.customer.file.%{s
ource node}") FORMAT DEL
;
-- Select statement specifies customer_demographics table
SELECT * FROM BCUAIX.CUSTOMER_DEMOGRAPHICS;
OUTFILE("%{source_host}:/work%{source_node}/HPU/bcuaix.demographics.file
.%{source_node}") FORMAT DEL
;
```

The preceding example incorporates these recommendations:

- TAKEN AT

Use this parameter to specify a partial backup image timestamp to reference multiple backup images, including incremental and delta backup images.

For example, a timestamp of 20110630 will instruct the `db2hpu` command to look for all backup images on June 30th 2011. Where multiple full backup images contain the partial timestamp, HPU terminates and outputs a message to that multiple backup images with the same partial timestamp were found. To allow HPU to determine the most recent backup images to use, omit the TAKEN AT clause.

- ON SOURCE HOST

  Use this parameter to instruct HPU to unload data in parallel to individual output files on each database partition instead of creating a single large file on the administration node.

  This action increases parallelism in a shared-nothing architecture since each output file is created locally on each data node. Each data node involved recovers the data for its local DB2 database partitions to its local output path with the correct file per node specified by the OUTFILE key.

- OUTFILE with {source_host} and {source_node} keywords

  Use the OUTFILE("%{source_host}:/OUTPUTPATH/FILE.%{source_node}") keywords to generate the output file on the correct file system associated with each database partition.

  Each database partition has a separate output file created with the database partition number appended to the name of the file.

Creating a control file

Follow these recommendations when creating a control file for an unload operation:

- Identify the portion of data that needs to be recovered. This action speeds up the recovery by limiting the amount of data to be unloaded to the output paths.
- Specify the data to be recovered in the control file by using the WHERE clause to specify the range of values from columns, DATAPARTITION ID to reference particular table range, PART clause to reference particular database partitions. This method is suitable for a scenario where corruption on an individual database has occurred.
- Reference a database catalog backup image that is consistent with the backup image. For example, if you are unloading data for a dropped table you need to reference a database catalog that has the definition of that dropped table.
- Verify that the output paths specified in the control file for the staging and output file areas exist and are accessible to the DB2 instance *`userid`*.
- Throttle the resources used by HPU using the *nbcpu* configuration parameter to reduce the number of threads created to mitigate the effect of `db2hpu` on other workloads.
- Control the amount of memory to be used by HPU by using the *bufsize* parameter.

*Issuing the **db2hpu** command*

The `db2hpu` command is issued as shown in this example where ***–i*** is the db2 instance name and ***-f*** specifies the control file to be used:

```
db2hpu -i instname -f controlfile
```

The control file settings override the defaults set in the product configuration file, *db2hpu.cfg*, which is located in the installation directory, typically */opt*.

- Ensure that any backup, ingest, or other operations scheduled that might affect the unload process have been altered for the duration of the unload exercise. Avoid contention for tape drives and TSM resources by ensuring that no other operations are active at the time of running HPU.

- Ensure that you always have the latest backup of the database catalog reflecting any changes available on disk. Using an up-to-date catalog reduces the need for HPU to access the backup image via tape.

Unload data from backup images by following the same pattern that was used to create the backup images.

Installing and configuring HPU in an IBM Smart Analytics System

HPU is highly scalable and can employ all processor and network resources made available to unload data at high speeds. Assess the effect on your test environment before implementing HPU in a production environment.

Configure your output directories to allow HPU to unload data with a single command in parallel across multiple database partitions.

Installing HPU in an IBM Smart Analytics System

Install and configure HPU on each node where you intend to use the product. Avoid copying the installation package to each individual node by locating it on the shared `/db2home` file system.

There are no additional installation requirements outside the standard product installation instructions.

When installed in a DB2 V9.7 environment, HPU can unload data from a DB2 v9.5 backup image.

Configuring HPU in an IBM Smart Analytics System

Configuration of HPU consists of three tasks:

- Creating and sharing a single configuration file across all nodes
- Creating a directory structure for HPU staging and output files
- Setting configuration file parameters

Creating and sharing a single configuration file across all nodes

Minimize administration of configuration files by creating a shared configuration file on the administration node and modifying the local configuration file on each node to reference it. To create a single configuration file that is shared across all nodes on which HPU is installed:

1. Make a copy of the default configuration file, customize it as required and save it on a file system that is accessible across all nodes; `/db2home` is recommended.
2. Add the *dir_cfg* parameter to the default HPU configuration file on each data node where HPU is installed to reference the shared HPU configuration file.
3. Make further customizations to the referenced configuration file only.

Configure and share a single HPU configuration file across all nodes to avoid confusion when operating HPU.

Creating a directory structure for HPU staging and output files

A table in a table space that is 100 GB in size requires up to 100 GB of disk space for staging data, Additional disk space is required for staging data if incremental backup images are referenced.

Staging Area

You can specify only one directory for the staging area. It is recommended that you create the same directory structure on each data node to accommodate staging of data. Where possible, locate the output and staging directories for each node on a separate file system on separate logical unit number (LUNs). Size the staging area to accommodate the largest table space in your database and also take into account your incremental and delta backup policy.

Output Area

To enable parallelism when using HPU, it is recommended that you create an output directory for each database partition . Appendix A shows how to use symbolic links to simplify the use of HPU with the `LOAD` utility. Size the output area to accommodate the largest table space in your database.

On an IBM Smart Analytics System, the use of the **`/db2fs`** file system is **not** recommended for HPU output and staging areas as it might compromise the database containers. The options available on an IBM Smart Analytics System depend on the model installed (refer to your documentation for further information):

- The IBM Smart Analytics System 5600 can be configured during the build phase to assign a pre-determined amount of disk space to a **`/bkpfs/bculinux/NODE$N`** file system. This file system is configured to be on a separate logical unit number (LUN) from the **`/db2fs`** file system.
- The IBM Smart Analytics System 7700 R1.1 and 7600 can be configured during the build phase to assign a pre-determined amount of disk space to a **`/workN`** file system where **`N`** is the database partition number. This space is reallocated from the **`/db2fs`** file system.
- A **`/stage`** file system is available on the admin node and can be mounted via NFS or GPFS across the data nodes. This tactic is suitable for smaller systems only as the amount of space is limited and the network traffic generated in a recovery situation could be prohibitive in larger environments.

Where the system is already installed and space is available, consider reconfiguring the storage to facilitate the creation of a file system for HPU use. The configuration of the test system use is outlined in Appendix A and symbolic links are recommended to accommodate a more elegant solution.

Setting configuration file parameters

The product configuration file holds default parameter values that are referenced each time the HPU utility is used.

Recommendations apply to these parameters:

- nbcpu

  Determine the number of work threads to be initiated by assessing the recovery time objective required and the resources available. Change the configuration parameter as required before running the `db2hpu` command.

- stagedir

  Use separate directories for the staging area and output area where the load formatted output files are written. The *stagedir* parameter in the *db2hpu.cfg* configuration file is set to */tmp* by

default. Change this entry to reference a directory that exists on each data node and has enough disk space to accommodate your HPU usage strategy.

- bufsize

  Default memory usage of 8 MB per database partition is recommended for most implementations, which is calculated as 4 MB for each HPU bufferpool. Increasing this configuration parameter value increases the amount of memory reserved by HPU with only a marginal increase in throughput and is not recommended.

- maxselects

  Determine the number of tables to process in parallel based on testing and the amount of resources allocated to recovery.

HPU disk requirements

The staging area must be sized to accommodate the temporary files created when data is unloaded. The minimum size needed for the staging area depends on:

- The DB2 software version. In DB2 V9.7 the entire table space is staged up to the High Water Mark (HWM) for DMS table spaces. Use the "`ALTER TABLESPACE <TSNAME> REDUCE`" command in DB2 V9.7 to avoid unnecessary large HWM on DMS table spaces.
- The table space type. SMS table spaces do not require the entire table to be staged, staging is complete when all table initialization data has been found.

It is recommended that you allocate space equivalent to the size of the table space.

When multiple tables are unloaded in parallel, the HPU utility attempts to minimize the number of times the backup image is accessed. Additional processing is required for objects including LOB, XML, LONG, LONG VARCHAR, and LONG VARGRAPHIC and is discussed in Appendix E.

Conclusion

Use IBM InfoSphere Optim High Performance Unload as a tool to address recovery scenarios where traditional restore and rebuild options are not viable.

The following recommendations should be taken into account when using HPU in your environment:

- Install HPU only on the physical servers where you intend to run HPU to unload data.
- Retain a copy of the most recent database catalog backup image on disk to help increase recovery speed when using HPU.
- Use offline backup images or consistent online backup images to help ensure the integrity of the data unloaded.
- Choose your recovery method based on your resource priority: use named pipes where there is insufficient disk space on the data nodes, or unload to output files where processor usage is limited.
- Use the configuration and control file parameters to throttle HPU usage of processor resources.
- Avoid contention when unloading data from tape by ensuring that the sequence of database partitions referenced in the unload blocks mirrors the backup sequence.
- Unload data from backup images by following the same pattern that was used to create the backup images.
- Configure your output directories to allow HPU to unload data with a single command in parallel across multiple database partitions.
- Configure and share a single HPU configuration file across all nodes to avoid confusion when operating HPU.

Appendix A. Configuration of test system used

The test system used for this paper was an IBM Smart Analytics System E7100 that consisted of three servers; an administration module and two data modules. The administration module had a single database partition that supported the catalog function, the coordinator function and non-partitioned metadata, staging and data warehouse tables. Each data module had eight database partitions that contained partitioned data warehouse tables.

- InfoSphere Optim High Performance Unload for DB2 for Linux, UNIX, and Windows version 32 bits 04.01.003.01(100929)
- DB2 version: DB2 V9.7 with Fix Pack 3 installed.
- AIX version: 6.1.0.0

Two tape drives were available on the TSM server. For online table space backups, 2 database partitions were backed up at a time; starting with 1 and 9 followed by 2 and 10 and continuing in this sequence. Online tables space backups were coordinated with the data ingest application so that no transactions occurred on the tables during the online backup.

Staging Area

The directory **`/work1/hpu/stage`** was set in the configuration file. Symbolic links were used to ensure that this directory was available on each data node so that data both unloaded and staged on each data node. The staging area is a temporary area and HPU removes temporary files once the unload operation finished. The example below shows the commands used to create the symbolic links on the administration node and the second data node. The first data node already contained the directories `/work1` through `/work8`. Note that these commands were used for all database partitions on each node.

```
ln –s /work0 /work1

ln –s /work9 /work1

mkdir /work1/hpu

mkdir /work1/hpu/stage
```

Output Area

The **/workN** file system was used to contain HPU output files local to each database partition. Because the **`HPU %{source_node}`** parameter denotes a database partition number as **`NODE0000`**, symbolic links were created to reference the **`/workN`** directory structure. Directories were created as follows:

```
ln –s /work1 /work001

mkdir /work1/HPU

ln –s /work2 /work002

mkdir /work2/HPU

ln –s /work3 /work003

mkdir /work3/HPU
```

```
..
ln –s /work9 /work009
mkdir /work9/HPU
ln –s /work16 /work016
mkdir /work16/HPU
```

To facilitate parallel load of the output files created in the directories shown above the following symbolic links were created on the second data node so that each data node has **/work1** through **/work8**:

```
ln –s /work9 /work1
ln –s /work10 /work2
ln –s /work11 /work3
..
ln –s /work16 /work8
```

To facilitate the unloading of data from backup images on disk the following directory structure was created to contain the backup images.

```
mkdir /work1/backup
mkdir /work2/backup
mkdir /work3/backup
..
```

The backup strategy in place was to back up a table space on one database partition per node in parallel until all database partitions on which the table space resided were backed up.

The HPU configuration file on the test system used was as follows:

```
# HPU default configuration
bufsize=4194304
db2dbdft=TPCDS
db2instance=BCUAIX
doubledelim=binary
netservice=db2hpudm412
#stagedir=/tmp
stagedir=/work1/hpu/stage
nbcpu=8
maxselects=10
```

Appendix B. Recovery scenario control file examples

The scenarios and example control files in this section illustrate how HPU can form a part of your recovery strategy using the recommendations made in this paper. The examples illustrate the recovery of partitioned tables through backup images stored on TSM. These examples cover:

- Recover dropped partitioned table using backup images on TSM
- Recover existing partitioned table using backup images on disk
- Recover dropped partitioned table using backup images on TSM and named pipes
- Recover dropped table space using backup images on TSM
- Recover data using non-production system and backup images on TSM

The backup strategy for each table space was to back up one database partition per node in parallel until the table space resided was backed up. In the examples below the same sequence was applied when unloading data.

Recover dropped partitioned tables using backup images on TSM

This scenario used HPU to recover two partitioned tables, `CUSTOMER` and `CUSTOMER_DEMOGRAPHICS` from online table space backup images stored on TSM. The data was unloaded to output files on each data node and then the DB2 Load utility was called as a separate step to load the data into the target table.

The output location on disk for each database partition was specified as `/workN/HPU`. An excerpt from the control file used in this scenario follows:

```
-- Global Block specifying backup image on disk where catalog resides
GLOBAL USING BACKUP CATALOG TPCDS FROM "/work0/HPU/catalog_BACKUP"
 TAKEN AT 2011033108;
;

-- Unload Block specifies backup image for partitions 1 and 9
UNLOAD TABLESPACE
PART(1,9)
USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM ON SOURCE
 HOST TAKEN AT 20110331080240;

-- Select Blocks specifies customer table and output files
SELECT * FROM BCUAIX.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/bcuaix.customer.file.%{
source_node}") FORMAT DEL

-- Select Blocks specifies customer_demographics table and output files
```

```
SELECT * FROM BCUAIX.CUSTOMER_DEMOGRAPHICS;
OUTFILE("%{source_host}:/work%{source_node}/HPU/bcuaix.demographics.fil
e.%{source_node}") FORMAT DEL
;
-- Unload Block specifies backup image for partitions 2 and 10
UNLOAD TABLESPACE
PART(2,10)
USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM ON SOURCE
  HOST TAKEN AT 20110331084220;
SELECT * FROM BCUAIX.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/bcuaix.customer.file.%{
source_node}") FORMAT DEL

SELECT * FROM BCUAIX.CUSTOMER_DEMOGRAPHICS;
OUTFILE("%{source_host}:/work%{source_node}/HPU/bcuaix.demographics.fil
e.%{source_node}") FORMAT DEL
;
-- NB: Repeat sequence for database partitions 3 & 11 and on to 8 & 16.
```

The data was unloaded in sequence according to the physical tape drives available. The following **LOAD** commands were used to load the data files created by the HPU process into the target partitioned table in parallel with no network traffic generated. Not all database partitions are included in the example:

```
db2 "LOAD from bcuaix.customer.file OF DEL INSERT
      INTO BCUAIX.CUSTOMER PARTITIONED DB CONFIG MODE
      LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work1/HPU
      OUTPUT_DBPARTNUMS(1,9)"

db2 "LOAD from bcuaix.customer.file OF DEL INSERT
      INTO BCUAIX.CUSTOMER PARTITIONED DB CONFIG MODE
      LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work2/HPU
      OUTPUT_DBPARTNUMS(2,10)"
-- Issue LOAD commands for all database partitions to be recovered
```

```
db2 "LOAD from bcuaix.demographics.file OF DEL INSERT INTO
      bcuaix.customer_demographics PARTITIONED DB CONFIG MODE
      LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work1/HPU
      OUTPUT_DBPARTNUMS(1,9)"

db2 "LOAD from bcuaix.demographics.file OF DEL INSERT INTO
      bcuaix.customer_demographics PARTITIONED DB CONFIG MODE
      LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work2/HPU
      OUTPUT_DBPARTNUMS(2,10)"
-- Issue LOAD commands for all database partitions to be recovered
```

Use the LOAD_ONLY_VERIFY_PART mode instead of LOAD_ONLY mode as HPU does not generate the headers with partition number, partition map, and partitioning key columns which LOAD_ONLY mode requires. The appropriate headers were generated only for HPU repartitioning and HPU migration.

Recover existing partitioned tables using backup images on disk

This scenario used HPU to recover a partitioned table from online table space backup images stored on disk. This scenario applies where the table exists in the database but the contents have been compromised or corruption has occurred.

The unload block specified each path where backup images files exist. Because the table to be recovered existed in the database, the database catalog is referenced in the global control block rather than a catalog associated with a backup image. This action reduced the number of backup images to be read and therefore increased the speed of the HPU operation. The control file used in this scenario was:

```
-- Global Block uses local database catalog
GLOBAL CONNECT TO TPCDS;

-- Unload Block specifies backup images for partitions 1 - 16
UNLOAD TABLESPACE
PART(1:16)
 USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") FROM
"/work1/backup","/work2/backup","/work3/backup","/work4/backup",
"/work5/backup","/work6/backup","/work7/backup","/work8/backup"
 ON SOURCE HOST TAKEN AT 2011033108;

-- Select Blocks specifies customer table and output files
SELECT * FROM BCUAIX.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/bcuaix.customer.file.%{
source_node}") FORMAT DEL
;
```

The **LOAD** command issued:

```
db2 "LOAD from bcuaix.customer.file OF DEL INSERT INTO BCUAIX.CUSTOMER
      PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
      PART_FILE_LOCATION /work1/HPU OUTPUT_DBPARTNUMS(1,9)"

db2 "LOAD from bcuaix.customer.file OF DEL INSERT INTO BCUAIX.CUSTOMER
      PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
      PART_FILE_LOCATION /work2/HPU OUTPUT_DBPARTNUMS(2,10)"

-- Issue LOAD commands for all database partitions to be recovered
```

Recover dropped partitioned table using backup images on TSM and named pipes

This scenario used HPU to recover partitioned tables from online table space backup images stored on TSM and, by using named pipes, simultaneously loaded the data into the target database table.

The following control file was created to use in unloading the table from backup images on TSM for database partitions 1 and 9 which were the first database partitions on each data node in the test environment. As the table had been dropped on the production database the catalog on the catalog backup image was referenced.

1. The control file was created. Each unload block processed two database partitions. The following example shows the first two unload blocks for database partitions 1, 9 and 2, 10. The pattern continued until all database partitions were processed.

```
-- Global block uses backup image of catalog on disk
GLOBAL USING BACKUP CATALOG TPCDS FROM "/work0/HPU/catalog_BACKUP"
 TAKEN AT 2011033108;
;
-- 1,9 specifies partitions 1 and 9
UNLOAD TABLESPACE PART(1,9)
USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM ON SOURCE
 HOST TAKEN AT 20110331080240;
SELECT * FROM BCUAIX.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/pipe.%{source_node}")
 FORMAT DEL
;
-- 2,10 specifies partitions 2 and 10 (continues to 8 and 16)
UNLOAD TABLESPACE PART(2,10)
USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM ON SOURCE
 HOST TAKEN AT 20110331084220;
SELECT * FROM BCUAIX.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/pipe.%{source_node}")
 FORMAT DEL
;
```

2. The named pipes were created on each data node as specified in the control file. The names in this example have the format: /workN/HPU/pipe.XXX where XXX is the appropriate database logical node number.

   The named pipes were created for the database partitions to be processed by the unload block in the control file. The commands used to create the named pipes in the test environment are illustrated as follows:

```
-- Data Node 1
mkfifo /work1/HPU/pipe.001
mkfifo /work2/HPU/pipe.002
mkfifo /work3/HPU/pipe.003
mkfifo /work4/HPU/pipe.004
mkfifo /work5/HPU/pipe.005
mkfifo /work6/HPU/pipe.006
mkfifo /work7/HPU/pipe.007
mkfifo /work8/HPU/pipe.008

-- Data Node 2 (Symbolic links for /work1 to /work8 created)
mkfifo /work1/HPU/pipe.009
mkfifo /work2/HPU/pipe.010
mkfifo /work3/HPU/pipe.011
mkfifo /work4/HPU/pipe.012
mkfifo /work5/HPU/pipe.013
mkfifo /work6/HPU/pipe.014
mkfifo /work7/HPU/pipe.015
mkfifo /work8/HPU/pipe.016
```

3. The HPU command specifying the required control file was executed:

```
db2hpu -f CUSTOMER_disk_pipes.ctl
```

4. DB2 Load operation was started:

```
db2 "LOAD from pipe OF DEL INSERT INTO BCUAIX.CUSTOMER PARTITIONED DB
      CONFIG MODE LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work1/HPU
      OUTPUT_DBPARTNUMS(1,9)"

db2 "LOAD from pipe OF DEL INSERT INTO BCUAIX.CUSTOMER PARTITIONED DB
      CONFIG MODE LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work2/HPU
      OUTPUT_DBPARTNUMS(2,10)"

-- Issue LOAD commands for all database partitions to be recovered
```

Recover dropped table space using backup images on TSM

This scenario used HPU to recover all the tables in a table space from backup images on TSM. The target tables did not have to use the same table space name.

The following control file unloads all the tables in the table space HASHTS. It also instructs the HPU process to use table space backup images. The `USING BACKUP CATALOG` keyword is needed as in this scenario the table space was dropped and so not visible in the database catalog any more.

```
-- Global Block
GLOBAL USING BACKUP CATALOG TPCDS FROM "/work0/HPU/catalog_BACKUP"
 TAKEN AT 20111117094257;
;

-- Unload Block 1,9
UNLOAD TABLESPACE HASHTS PART(1,9)
 OUTFILE("%{source_host}:/work%{source_node}/HPU/file.%{source_node}")
 FORMAT DEL

USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM ON SOURCE
 HOST TAKEN AT 20111117105244;
;

-- Unload Block (2,10)
UNLOAD TABLESPACE HASHTS PART(2,10)
 OUTFILE("%{source_host}:/work%{source_node}/HPU/file.%{source_node}")
 FORMAT DEL

USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM ON SOURCE
 HOST TAKEN AT 20111117105244;
;

-- continue process for other database partitions
```

Recover data using non-production system and backup images on TSM

In this scenario, the contents of a column in a production table had been compromised. Backup images of the production database were located on the TSM server. A non-production system was used to recover and prepare the data before it was reapplied to the production system.

The production environment contained two data nodes and a total of 16 database partitions. The non-production system contained one data node and eight database partitions. There were two tape drives available.

HPU was installed and configured on the non-production system only and the non-production system was configured to use TSM. A copy of the backup image for the catalog partition on the production system was made available on the non-production system. Because only a single column was compromised, HPU was used to unload just the primary key and the contents of the affected column.

The control file for this scenario unloaded data from each backup image in sequence. The TSMNODE parameter in the control file was used to identify the correct TSM node where the backup image file exists.

The TARGET ENVIRONMENT and TARGET KEYS control file parameters direct output data for each target database partition to a separate file. The TARGET ENVIRONMENT parameter value specifies the instance name, database name, and the server name where the database catalog resides on the non-production system. The TARGET KEYS parameter value specifies the database partitions on which the table exists in the non-production system.

Each data node on the non-production system used `/workN/HPU` directory for the output files. The APPEND parameter in the OUTFILE clause ensured that each subsequent unload block did not overwrite the output data file already created.

```
-- Global Block specifying catalog partition backup image on disk
GLOBAL USING BACKUP CATALOG TPCDS from "/work0/HPU/catalog_BACKUP"
 TAKEN AT 2011033108;
;

-- Unload Block specifies backup image for database partition 1 and 2 -
-- from TSMNODE tsm_datanode1
UNLOAD TABLESPACE
PART(1,2)
USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM TSMNODE
 "tsm_datanode1" TAKEN AT 2011033109;
-- Select Blocks specifies customer table and output files
SELECT * FROM BCUAIX.CUSTOMER;
TARGET ENVIRONMENT (INSTANCE "bcuaix" ON "perfbwadm_test" IN TPCDS)
TARGET KEYS(CURRENT PARTS(1:8))
OUTFILE
```

```
("%{target_host}:/work%{target_node}/HPU/BCUAIX.CUSTOMER.file.%{target_
node}") FORMAT DEL
;

-- Unload Block specifies backup image for database partitions 9 and 10
-- from TSMNODE tsm_datanode2
UNLOAD TABLESPACE
PART(9,10)
USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") USE TSM TSMNODE
 "tsm_datanode2" TAKEN AT 2011033109;
-- Select Blocks specifies customer table and output files
SELECT * FROM BCUAIX.CUSTOMER;
TARGET ENVIRONMENT (INSTANCE "bcuaix" ON "perfbwadm_test" IN TPCDS)
TARGET KEYS(CURRENT PARTS(1:8))
OUTFILE
("%{target_host}:/work%{target_node}/HPU/BCUAIX.CUSTOMER.file.%{target_
node}" APPEND) FORMAT DEL
;
```

Data was loaded into non-production system:

```
db2 "LOAD from BCUAIX.CUSTOMER.file OF DEL INSERT INTO BCUAIX.CUSTOMER
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION
/work1/HPU OUTPUT_DBPARTNUMS(1)"

db2 "LOAD from BCUAIX.CUSTOMER.file OF DEL INSERT INTO BCUAIX.CUSTOMER
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION
/work2/HPU OUTPUT_DBPARTNUMS(2)"
-- Issue LOAD commands for all database partitions to be recovered
```

Data was unloaded from non-production system to files in preparation for transfer to the production system. Only the column to be recovered is specified in the select block.

```
GLOBAL CONNECT TO TPCDS;
UNLOAD TABLESPACE
PART(1:8)
select customer_address_id,ca_zip FROM BCUAIX.CUSTOMER where ca_zip
like 'NA%4';
OUTFILE("/work1/HPU/customer_recovered_data.csv") FORMAT DEL
;
```

The data was applied to the production system by first being loaded into a temporary table and then applied to the target table through an UPDATE statement.

Appendix C. LOB and XML objects

HPU supports Large Objects (LOB) and XML data through an intermediate step in the unload/load process.

If the table is located in an SMS table space, the data files containing the externalized data (LOB and XML or both) are entirely staged while reading the backup a first time. The backup is read a second time by the HPU operation to process the main data pages of the table, containing references to LOB and XML or both data storage. During this second pass, this main data file can be partially staged, until the internal records describing the table have been found.

If the table is located in a DMS table space, the main data and the externalized ones are entirely staged while reading the backup. This action can result in several table spaces being staged, if the LOB or XML data is stored in a separate table space. The unload procedure is started against the staged files.

Further reading

Visit the best practices page on developerWorks for papers on physical database design, data lifecycle management, database backup, compression, and many other topics:

http://www.ibm.com/developerworks/data/bestpractices/

Refer to the following links for additional information about topics discussed in this paper:

- IBM InfoSphere Optim High Performance Unload

  http://www-01.ibm.com/software/data/optim/high-performance-unload-db2-luw/

- DB2 Load Utility

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.dm.doc/doc/c0004616.html

- Building a recovery strategy for an IBM Smart Analytics System Data Warehouse

  http://www.ibm.com/developerworks/data/bestpractices/isasrecovery/index.html

Traditional dropped table recovery method

- (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.ha.doc/doc/t0006318.htmlFirefoxHTML\Shell\Open\Command

Contributors

Alice Ma

Client Technical Professional, Enterprise Content Management, IBM

Austin Clifford

DB2 Data Warehouse QA Specialist, IBM

Bill Minor

Information Management Tooling & Development, IBM

Dale McInnis

DB2 Availability Architect & Senior Technical Staff Member, IBM

Jaime Botella Ordinas

IT Specialist & Accelerated Value Leader, Software Group, IBM

Vincent Arrat

HPU Development Team

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 grant 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.

Without limiting the above disclaimers, IBM provides no representations or warranties regarding the accuracy, reliability or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any recommendations or techniques herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Anyone attempting to adapt these techniques to their own environment do so at their own risk.

This document and the information contained herein may be used solely in connection with the IBM products discussed in this document.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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.

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. If these and