

# IBM Db2: Investigating Automatic Storage Table Spaces and Data Skew

George Wangelien

Zachary Hoggard







# Investigating automatic storage table spaces and data skew

The scope of this IBM® Redpaper™ publication is to provide a high-level overview of automatic storage table spaces, table space maps, table space extent maps, and physically unbalanced data across automatic storage table space containers (that is, data skew). The objective of this paper is to investigate causes of data skew and make suggestions for how to resolve it.

This paper is for Database Administrators (DBAs) of IBM Db2®; the DBAs should have general Db2 knowledge and skills. The environment used for the creation of this document is Db2 Version 11.1, and an IBM AIX® operating system. This document is based on results of testing various scenarios.

## Data skew and physical data imbalance

Data is usually balanced equally across Db2 database table space containers.

Example 1 consists of one table space spread across four file systems: /db2fs1, /db2fs2, /db2fs3, and /db2fs4. The one table space is the only data on these file systems. Data skew occurs at the table space level and not at the file system level, so this example is only for illustrative purposes.

*Example 1 Balanced file systems; each file system is 3% used*

df -lg /db2fs*					
Filesystem	GB blocks	Used	Free	%Used	Mounted on
/dev/db2lv4	5.00	0.14	4.86	3%	/db2fs1
/dev/db2lv5	5.00	0.14	4.86	3%	/db2fs2
/dev/db2lv6	5.00	0.14	4.86	3%	/db2fs3
/dev/db2lv7	5.00	0.14	4.86	3%	/db2fs4

Various events can cause data to favor specific table space containers, causing the data to become unequally distributed. In Example 2, more data is being written to the table space containers on file systems /db2fs3 and /db2fs4. File systems /db2fs1 and /db2fs2 are 3% full, and /db2fs3 and /db2fs4 are 20% full, demonstrating *data skew*.

*Example 2 More data is written to the table space containers*

---

```
df -lg /db2fs*
```

Filesystem	GB	blocks	Used	Free	%Used	Mounted on
/dev/db2lv4	5.00		0.14	4.86	3%	/db2fs1
/dev/db2lv5	5.00		0.14	4.86	3%	/db2fs2
/dev/db2lv6	5.00		0.97	4.03	20%	/db2fs3
/dev/db2lv7	5.00		0.97	4.03	20%	/db2fs4

---

Table space containers for one automatic storage table space can become unbalanced for various reasons:

- ▶ Hardware problem
- ▶ File system holding a container reaches 100% capacity
- ▶ File system maximum supported file size is reached

A good idea is to investigate why the data has become skewed.

## Automatic storage table spaces, and system and database managed space

IBM Db2 documentation makes a distinction between system managed space (SMS), database managed space (DMS), and automatic storage table spaces. IBM documentation addresses automatic storage table spaces as *automatic storage table spaces* and addresses DMS table spaces as *DMS* or *database managed space*. When reading IBM documentation, differentiating between them is important because their behaviors differ.

In a DMS table space, the DBA controls the storage space containers. Therefore, Db2 commands can be issued to configure DMS table space containers. For example, the **ALTER TABLESPACE** command can be issued to ADD, DROP, BEGIN NEW STRIPE SET, and EXTEND/RESIZE DMS table space containers.

With automatic storage table spaces, storage is managed automatically by Db2. Because Db2 controls the table space containers, users cannot configure the containers. For example, the **ALTER TABLESPACE** command cannot be used to ADD, DROP, BEGIN NEW STRIPE SET, or EXTEND/RESIZE on automatic storage table space containers because Db2 is controlling space management. Storage groups were introduced in Db2 V10.1, are defined by the DBA, and list a named set of storage paths. Storage paths list the file systems (absolute path) assigned for use by the automatic storage table spaces. Table spaces are assigned directly to a storage path in pre-Db2 V10. An example that shows how to display storage paths is in “Automatic storage table spaces and storage groups” on page 8.

Automatic storage table spaces containers are assigned file names by Db2 and are visible if you run commands or functions, such as these:

- ▶ list tablespace containers for <tsID> (deprecated)
- ▶ get snapshot for tablespaces on <dbname>
- ▶ db2pd -d <dbname> -tablespace
- ▶ sysibm.snapcontainer view (deprecated)
- ▶ MON\_GET\_CONTAINER (table function)
- ▶ MON\_GET\_TABLESPACE (table function)

## Displaying automatic storage table space types

Automatic storage table spaces can be SMS or DMS. The scope of this document is for DMS automatic storage table spaces. The type of each table space can be determined by running `db2pd -db <dbname> -tablespace` or by querying the `MON_GET_TABLESPACE` table function. Example 3 shows the SQL to determine if the table space is using automatic storage and if the table space is SMS or DMS. A value of 1 in column `TBSP_USING_AUTO_STORAGE` indicates that the table space is using automatic storage. Column `TBSP_TYPE` indicates an SMS or DMS table space.

*Example 3 Determine if table space uses automatic storage and if SMS or DMS*

```
select tbasp_name,
       tbasp_id,
       tbasp_using_auto_storage,
       tbasp_type,
       tbasp_content_type
from table(mon_get_tablespace('',-1)) as t
order by tbasp_name
```

TBSP_NAME	TBSP_ID	TBSP_USING_AUTO_STORAGE	TBSP_TYPE	TBSP_CONTENT_TYPE
SYSCATSPACE	0			
SYSTOOLSPACE	4			
TEMPSPACE1	1			
USERSPACE1	2			
USERSPACE2	3			

## Table space maps

A *table space map* is the Db2 representation of an automatic storage or DMS table space that describes the logical to physical conversion of page locations in a table space.

When a DMS or automatic storage table space is created, the table space map is created, along with the header page, the object table EMP, an SMP extent, and object table data extent. Data is striped evenly across all of the table space containers until an individual container fills up.

The pages in an automatic storage table space are grouped into extents, based on the extent size. Space in the table space is allocated to a table, one extent at a time. As the table is populated and an extent becomes full, a new extent is allocated.

Each table object is stored separately and each object allocates new extents as needed. Each table object is also paired with an extent map, which describes all of the extents in the table space that belong to the table object.

Example 4 shows a table space map for userspace2 (the output is from `get snapshot for tablespaces on <dbname>` command). Table space maps are explained in more detail later in this document.

*Example 4 Table space map for userspace2 (output from "get snapshot" command)*

Range	Stripe	Stripe	Max	Max	Start	End	Adj.	Containers
Number	Set	Offset	Extent	Page	Stripe	Stripe		
[ 0]	[ 0]	0	815	8159	0	203	0	4 (0,1,2,3)

## Extents

An *extent* is a block of storage in a table space container. Extents are made up of pages. When a table space is created the extent size and page size can be specified.

## Extent map

An *extent map* is a listing of extents in a table space. Figure 1 shows nine extents (0 - 8) for this table space. Extent 0 is located in stripe 0, container 0. Extent 7 is located in stripe 2, container 1.

		Container		
		0	1	2
Stripe	0	0	1	2
	1	3	4	5
	2	6	7	
	3	8		

Figure 1 Extent map

Although ranges have not yet been explained, stripes 0 - 1 belong to range 0, stripe 2 belongs to range 1, and stripe 3 belongs to range 2.

## Extent size

The *extent size* determines how many pages each extent will hold and how many pages are written to a container before writing to the next container. When a table space is created, an extent size is assigned to it. The default extent size is 32 pages.

The extent size for a table space can be determined by using the following commands or function:

- ▶ list tablespace containers for <tsID> (deprecated)
- ▶ get snapshot for tablespaces
- ▶ db2pd -d <dbname> -tablespace
- ▶ sysibm.snapcontainer view (deprecated)
- ▶ MON\_GET\_TABLESPACE (table function)

Each table space is associated with a table space ID (0 - 5) as shown in Example 5. Table space userspace2 has a table space ID of 3 and an extent size of 10.

Example 5 Table space associated with a table space ID

```
db2pd -d <dbname> -tablespace
```

Tablespace Configuration:

Id	Type	Content	PageSz	ExtentSz	Auto	NumCntrs	MaxStripe	LastConsecPg	Name
0	DMS	Regular	4096	4	Yes	4	0	3	SYSCATSPACE
1	SMS	SysTmp	4096	2	Yes	4	0	1	TEMPSPACE1
2	DMS	Large	4096	2	Yes	4	0	1	USERSPACE1
3	DMS	Large	4096	10	Yes	4	0	1	USERSPACE2
4	DMS	Large	4096	4	Yes	4	0	3	SYSTOOLSPACE
5	SMS	UsrTmp	4096	4	Yes	4	0	3	SYSTOOLSTMPSPACE

## Table space containers

Db2 assigns names to the automatic storage table space's containers. Containers hold database data. Each container has an ID number associated with it. The example described here has four containers (Container ID 0 - 3) with multiple table spaces sharing the same file systems.

The **db2pd -d <dbname> -tablespace** command can be used to list the table space IDs and table space container IDs. In this example, each file system (/db2fs1, /db2fs2, /db2fs3, and /db2fs4) contains multiple table spaces. File system /db2fs1 has table space containers for table spaces SYSCATSPACE, SYSTOOLSPACE, SYSTOOLSTMPSPACE, TEMPSPACE1, USERSPACE1, and USERSPACE2.

The following SQL was issued to list the table space IDs, table space container IDs, and the container name. The output is shown in Example 6.

```
select substr(tbsp_name,1,20) as tablespace_name,
       substr(char(tbsp_id),1,5) as tbsp_id,
       substr(char(container_id),1,5) as container_id,
       substr(container_name,1,54) as container_name
from table(mon_get_container('-',-1)) as t
order by tbsp_name
```

*Example 6 Output from SQL against a test database*

TABSPACE_NAME	TBSP_ID	CONTAINER_ID	CONTAINER_NAME
SYSCATSPACE	0	0	/db2fs1/db2inst2/NODE0000/TESTDB/T0000000/C0000000.CAT
SYSCATSPACE	0	1	/db2fs3/db2inst2/NODE0000/TESTDB/T0000000/C0000001.CAT
SYSCATSPACE	0	2	/db2fs2/db2inst2/NODE0000/TESTDB/T0000000/C0000002.CAT
SYSCATSPACE	0	3	/db2fs4/db2inst2/NODE0000/TESTDB/T0000000/C0000003.CAT
SYSTOOLSPACE	4	0	/db2fs1/db2inst2/NODE0000/TESTDB/T0000004/C0000000.LRG
SYSTOOLSPACE	4	1	/db2fs3/db2inst2/NODE0000/TESTDB/T0000004/C0000001.LRG
SYSTOOLSPACE	4	2	/db2fs2/db2inst2/NODE0000/TESTDB/T0000004/C0000002.LRG
SYSTOOLSPACE	4	3	/db2fs4/db2inst2/NODE0000/TESTDB/T0000004/C0000003.LRG
SYSTOOLSTMPSPACE	5	0	/db2fs3/db2inst2/NODE0000/TESTDB/T0000005/C0000000.UTM
SYSTOOLSTMPSPACE	5	1	/db2fs4/db2inst2/NODE0000/TESTDB/T0000005/C0000001.UTM
SYSTOOLSTMPSPACE	5	2	/db2fs1/db2inst2/NODE0000/TESTDB/T0000005/C0000002.UTM
SYSTOOLSTMPSPACE	5	3	/db2fs2/db2inst2/NODE0000/TESTDB/T0000005/C0000003.UTM
TEMPSPACE1	1	0	/db2fs3/db2inst2/NODE0000/TESTDB/T0000001/C0000000.TMP
TEMPSPACE1	1	1	/db2fs4/db2inst2/NODE0000/TESTDB/T0000001/C0000001.TMP
TEMPSPACE1	1	2	/db2fs1/db2inst2/NODE0000/TESTDB/T0000001/C0000002.TMP
TEMPSPACE1	1	3	/db2fs2/db2inst2/NODE0000/TESTDB/T0000001/C0000003.TMP
USERSPACE1	2	0	/db2fs1/db2inst2/NODE0000/TESTDB/T0000002/C0000000.LRG
USERSPACE1	2	1	/db2fs3/db2inst2/NODE0000/TESTDB/T0000002/C0000001.LRG
USERSPACE1	2	2	/db2fs2/db2inst2/NODE0000/TESTDB/T0000002/C0000002.LRG
USERSPACE1	2	3	/db2fs4/db2inst2/NODE0000/TESTDB/T0000002/C0000003.LRG
USERSPACE2	3	0	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000000.LRG
USERSPACE2	3	1	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000001.LRG
USERSPACE2	3	2	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000002.LRG
USERSPACE2	3	3	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000003.LRG

## Stripes

A *stripe* is a contiguous number of extents spanning one or more containers without repeating. Figure 2 shows four stripes: stripes 0 - 3. Stripe 0 spans container 0,1,2.

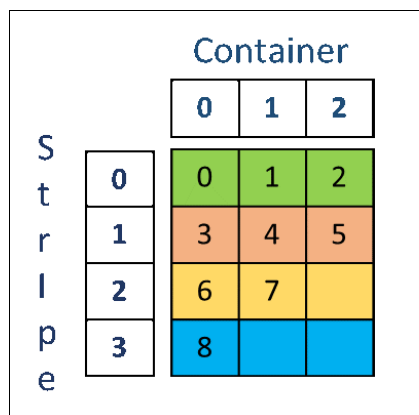


Figure 2 Stripes

## Ranges and stripe sets

A *range* is a contiguous number of stripes sharing the same common set of containers.

Figure 3 shows the following information:

- ▶ Range 0 – Stripe 0,1 – 6 extents
- ▶ Range 1 – Stripe 2 – 2 extents
- ▶ Range 2 – Stripe 3 – 1 extent

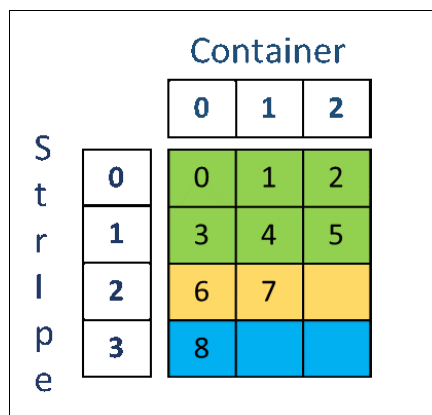


Figure 3 Range and stripe sets

Example 7 shows the table space map for Figure 3.

Example 7 Table space map for Figure 3

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[0]	[0]	0	5	59	0	1	0	3(0, 1, 2)
[1]	[0]	0	7	69	2	2	0	2(0, 1)
[2]	[0]	0	8	79	3	3	0	1(0)

A *stripe set* (which differs from a stripe) is a contiguous number of ranges. This example has one stripe set, which is stripe set 0.



Ranges and stripe sets; new stripe set added

Figure 4 and Example 8 show two ranges and two stripe sets.

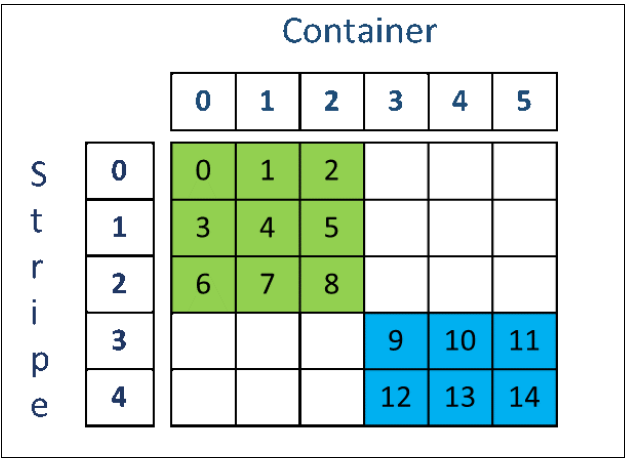


Figure 4 Ranges and stripe sets

Example 8 Ranges and stripe sets

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[0]	[0]	0	8	79	0	2	0	3(0, 1, 2)
[1]	[1]	0	14	139	3	4	0	3(3, 4, 5)

Table space map field definitions

Figure 5 shows an extent map, as described previously.

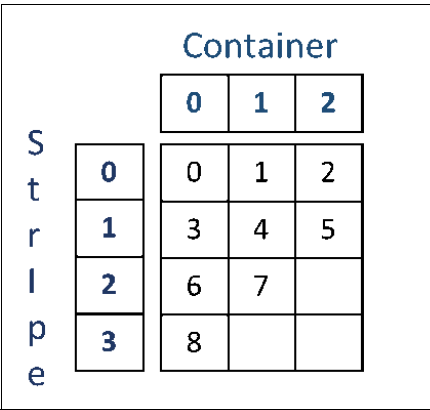


Figure 5 Extent map

Example 9 shows the table space map for the extent map.

Example 9 Table space map

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[0]	[0]	0	5	59	0	1	0	3(0, 1, 2)
[1]	[0]	0	7	69	2	2	0	2(0, 1)
[2]	[0]	0	8	79	3	3	0	1(0)

The table space map has the following fields:

<b>Range</b>	The range number, which always starts at zero (0).
<b>Stripe Set</b>	The stripe set number, which always starts at zero (0).
<b>Stripe Offset</b>	The extent number where the stripe set begins.
<b>Max Extent</b>	The maximum extent number found in the range.
<b>Max Page</b>	The maximum page number found in the range.
<b>Start Stripe</b>	The stripe number where the range starts in the map.
<b>End Stripe</b>	The stripe number where the range ends in the map.
<b>Adj</b>	The adjustment is the distance a range is shifted during a rebalance.
<b>Containers</b>	The containers that are part of the range. The container numbers are enclosed in parentheses and are preceded by the number of containers.

## Prerequisites for reclaiming storage using automatic storage table spaces

Having an associated storage group (or storage path) and having reclaimable space enabled are prerequisites for reclaiming storage using automatic storage table spaces.

### Automatic storage table spaces and storage groups

Storage groups were introduced in Db2 V10.1. Db2 levels prior to Db2 V10 (that is, Db2 V9.7) use a storage path. Storage groups and storage paths determine where the table space containers file systems reside. When a database with automatic storage is created, one storage group is created and named IBMSTOGROUP. The default storage group can be changed. Storage groups can be created, altered, or dropped with an **ALTER STOGROUP** command. When a table space is created, it will be assigned to the default storage group unless otherwise specified, and a table space may be assigned to only one storage group. A table space can change its assigned storage group during runtime, but the table space must be rebalanced after the reassignment.

The **db2pd** command can be used to display storage paths and storage groups. The output in Example 10 shows that storage paths are defined and the path state indicates InUse.

*Example 10 Output showing defined and InUse storage paths*

---

```
db2pd -db testdb -storagepaths OR db2pd -db testdb -storagegroups
Database Partition 0 -- Database TESTDB -- Active -- Up 0 days 02:32:06 -- Date
2017-02-15-20.18.43.590237
Storage Group Paths:
Address          SGID    PathID    PathState    PathName
0x07700000400F7F80 0        0        InUse        /db2fs1
0x07700000400F8320 0        1        InUse        /db2fs2
0x07700000400F86C0 0        2        InUse        /db2fs3
0x07700000400F8A60 0        3        InUse        /db2fs4
```

---

## Verify that table spaces are enabled for automatic storage and reclaimable space

You can reclaim storage only in table spaces created with Db2 Version 9.7 and later. Reclaimable storage is not available in table spaces created with earlier versions of Db2. You can see which table spaces in a database support automatic storage and reclaimable storage by using the `MON_GET_TABLESPACE` table function (Example 11). `MON_GET_TABLESPACE` will extract the `STORAGE_GROUP_NAME` if you are running Db2 V10 or later. The number 1 (one) in the column indicates that the feature is enabled.

*Example 11 Using the MON\_GET\_TABLESPACE function*

```
select varchar(tbsp_name,20) as tbsp_name,  
tbsp_id,  
tbsp_using_auto_storage,  
reclaimable_space_enabled  
-- storage_group_name  
from table(mon_get_tablespace('',-1)) as t  
order by tbsp_name
```

Output from the query is shown in Example 12.

*Example 12 Output*

TBSP_NAME	TBSP_ID	TBSP_USING_AUTO_STORAGE	RECLAIMABLE_SPACE_ENABLED
SYSCATSPACE	0	1	1
SYSTOOLSPACE	4	1	1
SYSTOOLSTMPSPACE	5	1	0
TEMPSPACE1	1	1	0
USERSPACE1	2	1	1
USERSPACE2	3	1	1

## Examples of data skew, rebalancing and reclaiming storage

For demonstration purposes, a test environment was created. It has a TESTDB database with a table space named USERSPACE2. The steps (labeled Step 1 through Step 8) in the remainder of this document demonstrate the progression of the test environment.

Db2 V11.1 is being used for this example.

### Step 1: Creating a database and an automatic storage table space

The following two commands were issued to create the database and automatic storage tablespace:

- ▶ `create db TESTDB automatic storage yes on /db2fs1, /db2fs2, /db2fs3, /db2fs4 dbpath on /restore_364/TESTDB_dbpath dft_extnt_sz 10`
- ▶ `create tablespace userspace2 autoresize yes increasesize 500 K`

## Step 2: Creating a table in the table space (userspace2)

The following command was issued to create the table:

- create table bigtable (c1 char(254),c2 char(254), c3 char(254), c4 char(254)) in userspace2

Table space userspace2 is located on these file systems: /db2fs1, /db2fs2, /db2fs3, /db2fs4. It has four containers, with these container IDs: 0, 1, 2, 3. Automatic storage table space containers are managed by Db2, but a user can still query them through any of these ways:

- Example of two IBM Db2 commands:
  - list tablespace containers for <tablespace\_id>, db2pd -d <dbname> -tablespaces
  - get snapshot for tablespaces on <dbname>
- Table function:
  - MON\_GET\_CONTAINER

The following SQL uses the MON\_GET\_CONTAINER table function to collect the container information. Only the output from userspace2 is shown (in Example 13).

```
select substr(char(tbsp_id),1,6) as tbsp_id,
       substr(tbsp_name,1,20) tablespace_name,
       substr(char(container_id),1,11) as container_id,
       -- container_name is varchar(256)
       substr(container_name,1,40) as container_name
from table(mon_get_container('',-1)) as t
order by tbsp_id, container_id
```

*Example 13 Output from userspace2*

TBSP_ID	TABLESPACE_NAME	CONTAINER_ID	CONTAINER_NAME
3	USERSPACE2	0	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000000.LRG
3	USERSPACE2	1	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000001.LRG
3	USERSPACE2	2	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000002.LRG
3	USERSPACE2	3	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000003.LRG

Use the following command to get a snapshot of userspace2 (Example 14):

get snapshot for tablespaces on TESTDB

*Example 14 Output of snapshot showing the data for userspace2*

Name	= USERSPACE2
Type	= Database managed space
Total pages	= 8200
Useable pages	= 8160
Used pages	= 50
Free pages	= 8110
High water mark (pages)	= 50
Page size (bytes)	= 4096
Extent size (pages)	= 10
Prefetch size (pages)	= 40
Number of containers	= 4

One extent is used by Db2 for each container: there are 4 containers so 4 extents are used by Db2 for a container tag. Because the extent size is 10, each extent holds 10 pages for a total of 40 pages. Total pages is 8200 = (8160 usable pages + 40 pages).

## Extent map for userspace2

The extent map for userspace2 is shown in Figure 6.

		Container			
		0	1	2	3
S t r i p e	0	0	1	2	3
	1	4	5	6	7
	2	8	9	10	11
	3	12	13	14	15
		Extents 16 thru 811 are not displayed			
	203	812	813	814	815

Figure 6 Extent map for userspace2

The output from the `get snapshot for tablespaces in <dbname>` command is shown in Example 15.

Example 15 Table space map for userspace2

Range	Stripe	Stripe	Max	Max	Start	End	Adj.	Containers
Number	Set	Offset	Extent	Page	Stripe	Stripe		
[ 0]	[ 0]	0	815	8159	0	203	0	4 (0,1,2,3)

There is one range consisting of stripes 0 - 203, for a total of 204 stripes: the first stripe starts on stripe 0 and the last stripe is 203.

Max Extent is 815 (extents 0 – 815), where 815 is the last extent, for a total of 816 extents.

Max Page is 8159. Pages are counted starting from page 0 so there are 8160 total pages. Each extent holds 10 pages, so 816 extents times 10 pages per extent equals 8160 usable pages.

## Step 3: Table space and extent map changes as data is added to the database

As data is added to the table space, more extents are allocated.

As Example 16 shows, the table space map for userspace2 has one table in it (bigtable). This is before additional data is loaded.

*Example 16 Table space map for userspace2 before additional data is loaded*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	815	8159	0	203	0	4 (0,1,2,3)

The following load command was issued several times to populate the table with data:

```
load from /backup/data.del of del savecount 100000 rowcount 100000 insert into bigtable nonrecoverable
```

The increase of data in the table space updates the table space map and extent map. Max Extent increased from 815 to 10019, as shown in Example 17.

*Example 17 Table space map for userspace2 after more data is loaded*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)

## Data skew concepts

Db2 uses striping to ensure an even distribution of data across the containers. This striping writes the data evenly across the containers in the table space, placing the extents for tables in round-robin fashion across all containers.

The storage paths that are defined by the storage group should all be of the same size. Having file systems of different sizes can result in uneven striping across the containers and can impact performance; for example, I/O parallelism might be affected. If any container is full, DMS table spaces use available free space from other containers.

Potential causes of data skew are as follows:

- ▶ Hardware problem occurs.
- ▶ Db2 database file system becomes full for these reasons:
  - Non Db2 database data is stored on the database file system.
  - System Administrator reduced the size of one of the table space container file systems.
  - Storage paths (file systems) belonging to one table space are not all the same size.
- ▶ Container reaches maximum file size permitted by the file system.

## Example: Data skew

This example consists of one table space spread across four file systems: /db2fs1, /db2fs2, /db2fs3, and /db2fs4. The one table space is the only data on these file systems. Data skew occurs at the table space level and not at the file system level, so this example is only for illustrative purposes.

Example 18 shows where (in our simulation) non Db2 data was placed on Db2 database file systems (/db2fs3, /db2fs4) causing two of the four Db2 database file systems to reach 100% capacity.

*Example 18 Two database file systems reach 100% capacity*

Filesystem	GB blocks	Used	Free	%Used	Mounted on
/dev/db2lv4	5.00	0.20	4.80	5%	/db2fs1
/dev/db2lv5	5.00	0.20	4.80	4%	/db2fs2
/dev/db2lv6	5.00	5.00	0.00	100%	/db2fs3
/dev/db2lv7	5.00	5.00	0.00	100%	/db2fs4

After the two file systems were at 100% capacity, users kept loading data into the database. But Db2 cannot add data into the containers on file systems /db2fs3 and /db2fs4 because they are full. Because two containers were full, the userspace2 table space used available free space from other containers (/db2fs1 and /db2fs2).

After the problem was discovered, the non Db2 data on the Db2 database file systems was removed, freeing up space. Db2 database file systems (/db2fs3 and /db2fs4) usage shows data is no longer equally balanced (Example 19).

*Example 19 Data is no longer equally balanced*

Filesystem	GB blocks	Used	Free	%Used	Mounted on
/dev/db2lv4	5.00	2.49	2.51	50%	/db2fs1
/dev/db2lv5	5.00	2.49	2.51	50%	/db2fs2
/dev/db2lv6	5.00	0.14	4.86	3%	/db2fs3
/dev/db2lv7	5.00	0.14	4.86	3%	/db2fs4

## Step 4: Table space and extent maps showing data skew

In the previous example (repeated here in Example 20), after the file systems on /db2fs3 and /db2fs4 reached 100% capacity, Db2 changed the table space map and extent map. In this next example, Db2 creates a new stripe set to remove the table space containers on /db2fs3 and /db2fs4, as shown in Example 21. Data is imbalanced because Db2 favors writing to container IDs 4 and 5 (/db2fs1 and /db2fs2).

*Example 20 Table space map before file systems are filled*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)

*Example 21 Table space map after file systems reached 100% capacity*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	133449	1334499	2505	64219	0	2 (4,5)

The extent map (Figure 7) and table space (Example 22) also show the results after the file systems reached 100% capacity.

		Container					
		0	1	2	3	4	5
S t r i p e	0	0	1	2	3		
	1	4	5	6	7		
	2	8	9	10	11		
	3	12	13	14	15		
		Extents 16 thru 10015 are not displayed					
	2504	10016	10017	10018	10019		
	2505					10020	10021
		Extents 10022 thru 133447 are not displayed					
	64219					133448	133449

Figure 7 Extent map of userspace2 after file systems reached 100% capacity

Example 22 Userspace2 table space after file systems reached 100% capacity

STRIPE_SET	CONTAINER_ID	CONTAINER_NAME
0	0	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000000.LRG
0	1	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000001.LRG
0	2	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000002.LRG
0	3	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000003.LRG
1	4	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000004.LRG
1	5	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000005.LRG

**Note:** A *stripe set* is a contiguous number of ranges. This example shows stripe set 0 (stripes 0 - 2504) and stripe set 1 (stripes 2505 - 64219).

The Db2 database data is no longer evenly striped across the userspace2 table space containers. Db2 removed /db2fs3 and /db2fs4 from stripe set 1. As data is added to the database, the data will become imbalanced across the containers. Rebalancing the table space containers will balance the data across the table space containers. See the next section (“Table space rebalancing concepts”).



## Table space rebalancing concepts

The process of *rebalancing* involves moving table space extents from one location to another to keep data evenly distributed across the table space containers.

The userspace2 table space map (Example 23) shows that data is not evenly striped across the table space containers. The table space will need to be rebalanced.

*Example 23 Data is not evenly striped*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	133449	1334499	2505	64219	0	2 (4,5)

## Example: How Db2 handles rebalancing operations

For illustrative purposes, here is a different example. It shows how Db2 handles free space during a table space rebalance operation.

Enough free space must be available on each file system in order to rebalance the table space. To determine how much space on the file system is needed to successfully rebalance, you must sum up all “missing containers.” For example, assume your table space map looks like the one in Example 24.

*Example 24 A sample table space map*

Range Number	Stripe Set	Containers
[ 0]	[ 0]	3 (0,1,2)
[ 1]	[ 1]	1 (3)
[ 2]	[ 2]	2 (4,5)

Rebalancing will create three extra Db2 containers (labeled as “N” in Example 25): two for stripe set 1 and one for stripe set 2, each equal in size to the container that already exists in a particular stripe.

*Example 25 Three extra containers*

Range Number	Stripe Set	Containers
[ 0]	[ 0]	3 (0,1,2)
[ 1]	[ 1]	1 (3,N,N)
[ 2]	[ 2]	2 (4,5,N)

If not enough free space is available for rebalancing to occur, rebalancing will fail. In our test with Db2 V11.1, the rebalancing returned the following error:

```
alter table space userspace2 rebalance
“SQL2094W The rebalance of tablespace <tablespace name> either did not add or
drop containers, or there was insufficient disk space to create all of the
containers. Reason code: <reason-code>.”
```

## Example: Rebalancing, before and after

The table space map before the table space was rebalanced is shown in Example 26.

*Example 26 Table space map before rebalancing*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	3347	33479	0	836	0	4 (0,1,2,3)
[ 1]	[ 1]	837	16688	166889	837	5283	0	3 (4,5,6)
[ 2]	[ 2]	5284	23357	233579	5284	7506	0	3 (7,8,9)
[ 3]	[ 3]	7507	30038	300389	7507	9733	0	3 (10,11,12)
[ 4]	[ 4]	9734	36708	367089	9734	13068	0	2 (13,14)

Containers are not distributed equally across the stripe sets (Example 27). There should be four containers per stripe set.

*Example 27 Distribution of containers is unequal across stripe sets*

CONT ID	STRIPE SET	TOTAL PAGES	USABLE PAGES	CONTAINER_NAME
0	0	8380	8370	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000000.LRG
1	0	8380	8370	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000001.LRG
2	0	8380	8370	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000002.LRG
3	0	8380	8370	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000003.LRG
4	1	44480	44470	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000004.LRG
5	1	44480	44470	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000005.LRG
6	1	44480	44470	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000006.LRG
7	2	22240	22230	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000007.LRG
8	2	22240	22230	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000008.LRG
9	2	22240	22230	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000009.LRG
10	3	22280	22270	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000010.LRG
11	3	22280	22270	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000011.LRG
12	3	22280	22270	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000012.LRG
13	4	33360	33350	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000013.LRG
14	4	33360	33350	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000014.LRG

Example 28 shows the table space map after the table space is rebalanced.

*Example 28 Table space map after rebalancing*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	3347	33479	0	836	0	4 (0,1,2,3)
[ 1]	[ 1]	837	21135	211359	837	5283	0	4 (4,5,6,19)
[ 2]	[ 2]	5284	30027	300279	5284	7506	0	4 (7,8,9,18)
[ 3]	[ 3]	7507	38935	389359	7507	9733	0	4 (10,11,12,17)
[ 4]	[ 4]	9734	52275	522759	9734	13068	0	4 (13,14,15,16)

Containers are now striped across the file systems (Example 29).

*Example 29 Containers now equally striped across file systems*

CONT ID	STRIPE SET	TOTAL PAGES	USABLE PAGES	CONTAINER_NAME
0	0	8380	8370	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000000.LRG
1	0	8380	8370	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000001.LRG
2	0	8380	8370	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000002.LRG
3	0	8380	8370	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000003.LRG
4	1	44480	44470	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000004.LRG
5	1	44480	44470	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000005.LRG
6	1	44480	44470	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000006.LRG
19	1	44480	44470	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000019.LRG
7	2	22240	22230	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000007.LRG
8	2	22240	22230	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000008.LRG
9	2	22240	22230	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000009.LRG
18	2	22240	22230	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000018.LRG
10	3	22280	22270	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000010.LRG
11	3	22280	22270	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000011.LRG
12	3	22280	22270	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000012.LRG
17	3	22280	22270	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000017.LRG
13	4	33360	33350	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000013.LRG
14	4	33360	33350	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000014.LRG
15	4	33360	33350	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000015.LRG
16	4	33360	33350	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000016.LRG

## Step 5: Estimating free disk space required for rebalancing

This step uses the example to estimate the space required for rebalancing. For userspace2, Example 30 shows the current table space map; Example 31 shows the snapshot.

*Example 30 Current table space map for userspace2*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	133449	1334499	2505	64219	0	2 (4,5)

*Example 31 Snapshot of userspace2*

Container ID	= 4
Total Pages in Container	= 617160
Usable Pages in Container	= 617150
Stripe Set	= 1
Container ID	= 5
Total Pages in Container	= 617160
Usable Pages in Container	= 617150
Stripe Set	= 1

The rebalancing will create two more new containers in stripe set 1. The containers will be equal in size to the already existing containers in stripe set 1 (4,5).

Containers 4 and 5 each have 617150 pages of Usable Pages in Container and have a total of 1234300 usable pages (617150 + 617150). Rebalancing will create two new containers of 617150 usable pages each.

Another way to calculate Usable Pages in Container is by using only the table space map. The Max Page value in stripe set 1 minus the Max Page value in stripe set 0 equals total pages in stripe set 1. Total usable pages in stripe set 1 is 1234300, which equals 1334499 minus 100199. There are two containers in stripe set 1 so Usable Pages in each container is:

$$617150 = 1234300 / 2$$

Db2 is using an additional extent per container. Extent size is 10 so 10 more pages must be added to calculate Total Pages in Container, which is 617160 4K-pages (size of each container).

Given a page size of 4K, 617160 pages equals 2411 MB or about 2.4 GB. Rebalancing will create new containers, which are missing from stripe set 1. The file systems where the containers will be created must have enough free disk space. To create the missing containers, each file system will need more than 2.4 GB of free disk space. Additional disk space on the file system must be allocated to be safe.

If you allocate only the minimum space required after the rebalance completes, the file systems for the two missing containers will be near or at 100% full. Also, a risk exists that rebalancing might fail because of insufficient disk space. Extra disk space needs to be allocated only on a temporary basis, because space can be freed up when storage is reclaimed in the next step.

Stripe set 1 must have four containers defined to keep the data balanced. Stripe set 1 is missing two containers. To determine which containers are missing, check the storage path being used for the table space to determine what file systems the stripe set is using. Use the **db2pd** utility to correlate the existing container IDs (4,5) to the file system. In this example, the storage path is using /db2fs1, /db2fs2, /db2fs3, and /db2fs4. The container IDs (4,5) map to file systems /db2fs1 and /db2fs2. Rebalancing will create a new container on /db2fs3 and /db2fs4.

Before the rebalance is run, note that containers 4 and 5 have a total of 1234300 usable pages (617150 + 617150). The table space map shows Max Page = 1334499. Adding all the pages in the table space map (Max Page in stripe set 0 plus Usable pages in containers 4,5) will equal the Max Page in stripe set 1:

$$100199 + 617150 + 617150 = 1334499 \text{ Max Page}$$

## Insufficient free disk space for rebalancing to create new containers

In this test scenario, not enough free disk space is allocated for rebalancing. Example 32 shows the current map.

*Example 32 Current table space map for userspace2*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	133449	1334499	2505	64219	0	2 (4,5)

With that table space map the rebalance command will create two new containers in stripe set 1. Sufficient free disk space must be available on the file systems to create the two new containers. If not enough disk space is available on the file system (or file systems) to create

the containers, then the **rebalance** command will generate an error message, as in the following example:

SQL2094W The rebalance of table space <table space name>" either did not add or drop containers, or there was insufficient disk space to create all of the containers. Reason code: <reason-code>.

If a file system does not have enough free disk space to create the container, the rebalance will attempt to use available free space from other file systems. If enough free space is available on the other file systems, the rebalance will still run, but the data will not be correctly balanced because containers will be missing, because they could not be created.

To create a test environment to demonstrate that situation, a large file was placed on file system /db2fs4 (container 3). File system /db2fs4 (container 3) is now 100% full so no free disk space is available for the rebalance to create another container on /db2fs4.

The **rebalance** command was executed and it returned the SQL2094 error message because a new container could not be created on /db2fs4.

The **rebalance** command continued to run because enough free disk space existed across both stripe sets even though the container on /db2fs4 could not be created. The other file systems had enough free space for the rebalance to run. Data could be rebalanced across only three of the file systems in stripe set 1. Data is still skewed after rebalancing completes. Example 33 shows the map after rebalancing.

*Example 33 Table space map for userspace2 after rebalance*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	195164	1951649	2505	64219	0	3 (4,5,6)

## Step 6: Rebalancing the table space

**Important:** The **rebalance** command can cause a significant performance impact.

Consider the following information before you rebalance a table space:

- ▶ Be sure you have a back out plan in case of problems.
- ▶ The Db2 command options for the **rebalance** command differ among Db2 versions. Check the IBM Db2 documentation to determine if the **rebalance** command can be throttled, suspended, or resumed for your version of Db2.
- ▶ Ensure you fully understand the implications of rebalancing table spaces. Before you attempt rebalancing, see the IBM Db2 documentation for more information about rebalancing table spaces.
- ▶ Changes involving rebalancing table spaces must be carefully planned and communicated.
- ▶ If possible, test rebalancing on a test server and benchmark the performance.

### Rebalance command

To rebalance, use the following command (Db2 V11.1):

```
alter tablespace userspace2 rebalance
```

## Monitoring the rebalance process

To monitor the rebalance, see the output from these commands and table function:

- ▶ The **get snapshot for tablespaces on testdb** command (Example 34)
- ▶ The **list utilities show detail** command (Example 35)
- ▶ The **MON\_GET\_REBALANCE\_STATUS** table function (Example 36)

*Example 34 Output from the “get snapshot for tablespaces on testdb” command*

---

Tablespace State	= 0x'10000000'
Detailed explanation:	
DMS rebalancer is active	
Rebalancer Mode	= Forward
Start Time	= 02/18/2017 23:00:53.000000
Restart Time	= 02/18/2017 23:00:53.000000
Number of extents processed	= 44539
Number of extents remaining	= 88902
Last extent moved	= 54559
Current priority	= 0

---

*Example 35 Output from the “list utilities show detail” command*

---

ID	= 1429
Type	= REBALANCE
Database Name	= TESTDB
Partition Number	= 0
Description	= Tablespace ID: 3
Start Time	= 03/10/2017 13:17:32.170315
State	= Executing
Invocation Type	= User
Throttling:	
Priority	= Unthrottled
Progress Monitoring:	
Estimated Percentage Complete	= 8
Total Work	= 133441 extents
Completed Work	= 10113 extents
Start Time	= 03/10/2017 13:17:32.170331

---

*Example 36 Output from MON\_GET\_REBALANCE\_STATUS table function*

---

```
select
  char(tbsp_name,1,30) as tbsp_name,
  dbpartitionnum,
  member,
  rebalancer_mode,
  rebalancer_status,
  rebalancer_extents_remaining,
  rebalancer_extents_processed,
  rebalancer_start_time
from table(mon_get_rebalance_status(NULL,-2)) as t
```

TBSP_NAME	DBPARTITIONNUM	MEMBER	REBALANCER_MODE	REBALANCER_STATUS
USERSPACE2	0	0	FWD_REBAL	ACTIVE

REBALANCER_EXTENTS_REMAINING	REBALANCER_EXTENTS_PROCESSED	REBALANCER_START_TIME
88902	44539	2017-01-18-23.00.53.000000

---

## Rebalance completed successfully

The table space map of userspace2 *before* rebalancing is shown in Example 37.

*Example 37 Table space map before rebalancing*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	133449	1334499	2505	64219	0	2 (4,5)

The table space map of userspace2 *after* rebalancing is shown in Example 38. Rebalancing added two containers (6,7). Data is now equally distributed across database containers as shown in Example 39.

*Example 38 Table space map after rebalancing*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	256879	2568799	2505	64219	0	4 (4,5,6,7)

*Example 39 Equal distribution across containers*

CONT ID	STRIPE SET	TOTAL PAGES	USABLE PAGES	CONTAINER NAME
0	0	25060	25050	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000000.LRG
1	0	25060	25050	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000001.LRG
2	0	25060	25050	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000002.LRG
3	0	25060	25050	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000003.LRG
4	1	617160	617150	/db2fs1/db2inst2/NODE0000/TESTDB/T0000003/C0000004.LRG
5	1	617160	617150	/db2fs2/db2inst2/NODE0000/TESTDB/T0000003/C0000005.LRG
6	1	617160	617150	/db2fs3/db2inst2/NODE0000/TESTDB/T0000003/C0000006.LRG
7	1	617160	617150	/db2fs4/db2inst2/NODE0000/TESTDB/T0000003/C0000007.LRG

## Rebalancing messages in db2diag.log (DIAGLEVEL 3 – Db2 V11.1)

Rebalancing messages in the db2diag.log file are shown in Example 40.

*Example 40 Rebalancing messages in db2diag.log (DIAGLEVEL 3 – Db2 V11.1)*

```

2017-02-18-23.00.53.465156-300 E1132378A396          LEVEL: Info
PID      : 18022412          TID   : 14131          PROC  : db2sysc 0
INSTANCE: db2inst2          NODE   : 000
EDUID    : 14131             EDUNAME: db2rebal (TESTDB) 0
FUNCTION: DB2 UDB, buffer pool services, sqlb_fwd_rebalance, probe:2204
MESSAGE  : ADM6058I Rebalancer for table space "userspace2" (ID "3") was
              started.

2017-02-18-23.09.30.692875-300 E1133268A398          LEVEL: Info
PID      : 18022412          TID   : 14131          PROC  : db2sysc 0
INSTANCE: db2inst2          NODE   : 000
EDUID    : 14131             EDUNAME: db2rebal (TESTDB) 0
FUNCTION: DB2 UDB, buffer pool services, sqlb_rebalance, probe:2876
MESSAGE  : ADM6062I Rebalance for table space "userspace2" (ID "3") has been
              completed.

2017-02-18-23.09.30.693188-300 I1133667A369          LEVEL: Warning

```

```

PID      : 18022412          TID   : 14131          PROC  : db2sysc 0
INSTANCE: db2inst2          NODE   : 000
EDUID    : 14131            EDUNAME: db2rebal (TESTDB) 0
FUNCTION: DB2 UDB, buffer pool services, sqlb_rebalance, probe:2876
DATA #1 : String, 39 bytes
PoolID 3: Last extent moved was #133440

```

---

## Reclaimable storage

*Reclaimable storage* is a feature of automatic storage table spaces and DMS table spaces. Temporary table spaces are not supported. You can use reclaimable storage to consolidate in-use extents below the high-water mark and return unused extents in your table space to the system for reuse.

## Reclaimable storage concepts

Db2 will attempt to lower the high-water mark and reduce the size of the automatic storage table space containers. The *high-water mark* refers to the page number of the first page in the extent following the last allocated extent. For example, if a table space has 1000 pages and an extent size of 10, there are 100 extents. If the 52nd extent is the highest allocated extent in the table space that means the high-water mark is 520. Lowering the high-water mark will enable Db2 to be able to reclaim storage.

Consider the following information about reclaimable storage:

- ▶ Be sure you have a back-out plan in case of problems.
- ▶ The **alter tablespace rebalance** command is asynchronous, online, not logged, and can be throttled.
- ▶ The **alter tablespace reduce** command, which can also be used to reclaim storage, is online, asynchronous, logged, and can be stopped in the middle of the operation.
- ▶ Ensure you fully understand the implications of reclaiming storage before doing it. See the IBM Db2 documentation for more information about reclaimable storage table spaces before attempt to reclaim storage.

After table space storage was reclaimed, space on the file system was reduced from 50% used space to 27% used space. See Example 41 through Example 43 on page 23; also see Figure 8.

*Example 41 File system space after rebalance was issued*

Filesystem	GB blocks	Used	Free	%Used	Mounted on
/dev/db2lv4	5.00	2.49	2.51	50%	/db2fs1
/dev/db2lv5	5.00	2.49	2.51	50%	/db2fs2
/dev/db2lv6	5.00	2.49	2.51	50%	/db2fs3
/dev/db2lv7	5.00	2.49	2.51	50%	/db2fs4

*Example 42 File system space after table space storage was reclaimed*

Filesystem	GB blocks	Used	Free	%Used	Mounted on
/dev/db2lv4	5.00	1.32	3.68	27%	/db2fs1
/dev/db2lv5	5.00	1.31	3.69	27%	/db2fs2
/dev/db2lv6	5.00	1.31	3.69	27%	/db2fs3
/dev/db2lv7	5.00	1.31	3.69	27%	/db2fs4



*Example 43 The userspace2 table space map after rebalance and before storage was reclaimed*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	256879	2568799	2505	64219	0	4 (4,5,6,7)

S  
t  
r  
i  
p  
e

Container								
0	1	2	3	4	5	6	7	
0	0	1	2	3				
1	4	5	6	7				
2	8	9	10	11				
3	12	13	14	15				
	Extents 16 thru 10015 are not displayed							
2504	10016	10017	10018	10019				
2505					10020	10021	10022	10023
2506					10024	10025	10026	10027
	Extents 10029 thru 256871 are not displayed							
64218					256872	256873	256874	256875
64219					256876	256877	256878	256879

*Figure 8 The userspace2 extent map after rebalance and before storage was reclaimed*

## Step 7: Reclaiming storage by running the alter tablespace command

You can reclaim storage by using any of these commands in Db2 V11.1:

- ▶ alter tablespace userspace2 lower high water mark
- ▶ alter tablespace userspace2 reduce
- ▶ alter tablespace userspace2 reduce max

Example 44 and Figure 9 on page 24 show userspace2 after storage is reclaimed.

*Example 44 The userspace2 table space map after storage is reclaimed*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	133443	1334439	2505	33360	0	4 (4,5,6,7)

		Container							
		0	1	2	3	4	5	6	7
S t r i p e	0	0	1	2	3				
	1	4	5	6	7				
	2	8	9	10	11				
	3	12	13	14	15				
		Extents 16 thru 10015 are not displayed							
	2504	10016	10017	10018	10019				
	2505					10020	10021	10022	10023
		Extents 10024 thru 133435 are not displayed							
	3360					133436	133437	133438	133439

Figure 9 The userspace2 extent map after storage is reclaimed

## Reclaimable storage messages in db2diag.log (DIAGLEVEL 3 – Db2 V11.1)

Reclaimable storage messages in the db2diag.log file are shown in Example 45.

*Example 45 Reclaimable storage messages in db2diag.log (DIAGLEVEL 3 – Db2 V11.1)*

---

```

2017-02-22-11.51.09.943766-300 E1572818A482      LEVEL: Info
PID      : 18022412          TID   : 13667      PROC  : db2sysc 0
INSTANCE: db2inst2          NODE   : 000        DB   : TESTDB
APPHDL   : 0-1951           APPID: *LOCAL.DB2.170222165109
AUTHID   : AODDBA
EDUID    : 13667            EDUNAME: db2agent (TESTDB) 0
FUNCTION: DB2 UDB, buffer pool services, sqlbExtentMovementEntryPoint, probe:4932
DATA #1 : <preformatted>
Extent Movement started on tablespace 3

2017-02-22-11.51.10.060161-300 E1573301A527      LEVEL: Info
PID      : 18022412          TID   : 13667      PROC  : db2sysc 0
INSTANCE: db2inst2          NODE   : 000        DB   : TESTDB
APPHDL   : 0-1951           APPID: *LOCAL.DB2.170222165109
AUTHID   : AODDBA
EDUID    : 13667            EDUNAME: db2agent (TESTDB) 0
FUNCTION: DB2 UDB, buffer pool services, sqlbLockAndMoveExtents, probe:4461
MESSAGE  : ADM6008I Extents within table space "userspace2" (ID "3") have been
            moved. Reason code = "0".

```

---

## Checking table spaces for data skew

To check for table spaces that are candidates for data skew, you can use a table space snapshot. Issue a snapshot for table spaces and scan the table space maps, looking for containers that are not balanced. If you see an unequal number of containers across different ranges, consider investigating further.

You can also use the `MON_GET_TABLESPACE_RANGE` table function. You can tailor the SQL (Example 46) to meet your requirements.

*Example 46 SQL with MON\_GET\_TABLESPACE\_RANGE function*

```
select
substr(tbsp_name, 1, 15) as tbsp_name,
range_number,
range_num_container
from table(MON_GET_TABLESPACE_RANGE(-1)) as t
group by  tbsp_name, range_number, range_num_container
```

Output for only userspace2 is shown in the next examples (Example 47 and Example 48 show balanced; Example 49 and Example 50 show unbalanced).

*Example 47 SQL output: A balanced table space*

TBSP_NAME	RANGE_NUMBER	RANGE_NUM_CONTAINER
-----	-----	-----
USERSPACE2	0	4
USERSPACE2	1	4

*Example 48 Table space map of userspace2 (tbsp\_id=3)*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	133443	1334439	2505	33360	0	4 (4,5,6,7)

*Example 49 SQL output: An unbalanced table space*

TBSP_NAME	RANGE_NUMBER	RANGE_NUM_CONTAINER
-----	-----	-----
USERSPACE2	0	4
USERSPACE2	1	2

*Example 50 Table space map of userspace2 (tbsp\_id=3)*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	10019	100199	0	2504	0	4 (0,1,2,3)
[ 1]	[ 1]	2505	133449	1334499	2505	64219	0	2 (4,5)

## Using the restore utility to fix data skew

Consider the following information:

- ▶ Rebalancing table space containers can significantly impact database performance. If you choose to rebalance, carefully plan the change.
- ▶ The Db2 restore utility can also be used to rebalance table space containers. You must follow certain conditions in order to ensure the restore utility redefines the automatic storage containers. If done properly, restoring will create balanced containers across one stripe set. A redirected restore can be used, with no redirection. For a successful change, see the IBM Db2 documentation to ensure all prerequisites of the restore utility are met.
- ▶ If a database has basically one large table space, restoring might be a better option than rebalancing the table space and reclaiming space. Some SAP databases fit this model.

## Resolving data skew by using the Db2 restore utility

Example 51 shows the userspace2 table space map with data skew.

*Example 51 Data skew*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	3347	33479	0	836	0	4 (0,1,2,3)
[ 1]	[ 1]	837	6693	66939	837	2509	0	2 (4,5)

The database was restored by using the following three commands:

- ▶ `restore database testdb tablespace userspace2 online from /backup taken at YYYYMMDDHHMMSS into testdb redirect`

Message SQL1277W indicates a redirected restore operation is being performed.

- During a table space restore, only table spaces being restored can have their paths reconfigured.
  - During a database restore, storage group storage paths and DMS table space containers can be reconfigured.
- ▶ `restore db testdb continue`
  - ▶ `rollforward db testdb to end of logs and complete`

The userspace2 table space map after restoring the database is shown in Example 52. The data skew issue is now corrected. The **restore** utility re-created the table space maps and extent maps.

*Example 52 Table space map after restoring the database*

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[ 0]	[ 0]	0	6719	67199	0	1679	0	4 (0,1,2,3)

## Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**George Wangelien** is a Senior Db2 UDB DBA for IBM Hybrid Cloud. George has over 22 years of Db2 for Linux, UNIX, and Windows experience at IBM. His interests range from consulting, problem determination, high availability solutions, proof of concepts, and writing ksh scripts. His areas of expertise are production Db2 for Linux, UNIX, and Windows support and also the administration of SAP Db2 databases. George holds numerous IT and Db2 certifications, a Db2 patent and a Bachelor's degree from Montclair State College, New Jersey, USA.

**Zachary Hoggard** is a Software Developer for IBM Db2 in the IBM Hybrid Cloud organization at the IBM Canada Lab. Zach has over 3 years of Db2 kernel development experience focusing on the buffer pool and storage management components. His interests range from helping customers with problem determination, developing new IBM Db2 pureScale® features, and high-performance processing. His areas of expertise are buffer pool and storage manager designs for pureScale and non pureScale instances. Zach is an IBM Certified Database Associate for Db2 v11.1 and v10.5, and holds a Bachelor's degree in Software Engineering from Western University in London, Ontario, Canada.

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:  
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:  
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new IBM Redbooks® publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>



# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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, MD-NC119, Armonk, NY 10504-1785, US*

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

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 actual people or business enterprises 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

Redbooks (logo) ®  
AIX®  
Db2®

IBM®  
pureScale®  
Redbooks®

Redpaper™

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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







REDP-5454-00

ISBN 0738456284

Printed in U.S.A.

Get connected

