Maximizing Performance of IBM DB2 Backups

This IBM® Redbooks® Analytics Support Web Doc describes how to maximize the performance of IBM DB2® backups.

Backing up a database is a critical part of any disaster recovery plan. Making sure that the backup completes in a reasonable time frame means that there is always a valid backup available in case of emergency. This document describes the following elements that are critical to good backup performance:

- The backup process model
- Basic backup performance concepts
- Backup performance considerations
- Monitoring backup performance with the DB2_BAR_STATS registry variable

This document applies to all versions of DB2 for Linux, UNIX, and Windows.

The backup process model

The backup process model consists of the engine dispatchable units (EDUs) db2agent, db2bm, and db2med. If the backup uses a storage manager, an additional db2vend process is created. The db2agent EDU does most of its work during the start of the backup, and then waits for the backup to complete. The tasks that it performs include the following actions:

- Calculate and create an optimal number of db2bm and db2med EDUs
- Allocate an optimal number of optimally sized backup buffers
- Display the selected tunable values to the db2diag.log
- Allocate message queues for inter-EDU communication
- Create a list of table spaces that are sorted by decreasing size
- Coordinate the backup process by sending control messages to the db2bm and db2med EDUs

The db2bm EDU acquires a backup buffer from the empty queue, and then fills the buffer by reading data from its assigned table space. The db2bm then places the backup buffer on the full queue to be written to the backup device by the db2med EDU.

The db2med EDU is responsible for acquiring a backup buffer from the full queue and writing it to the backup device. If the backup is being written to disk, it handles the input/output (I/O) directly. If the backup is being written to a storage manager, it transfers the full buffer to the db2vend process. After confirmation that the db2vend process successfully saved the contents of the buffer, the db2med EDU returns the backup buffer to an empty queue for reuse by the next db2bm EDU.

The db2vend process is created by the db2med EDU if a storage manager is being used. The db2vend process receives a full buffer from the db2med, and calls the appropriate vendor application programming interface (API) for sending the data to the storage manager. A separate process is used so that any crash or abnormal failure during interaction with the storage manager terminates only the db2vend process, and does not negatively affect the stability of the DB2 instance.
**Basic backup performance concepts**

The following concepts are critical to designing and implementing a successful backup strategy:

- **Balanced table spaces are the primary factor**
  
  The primary way to ensure good backup performance is to make sure that the data is evenly distributed between table spaces. During the backup, DB2 assigns one `db2bm` EDU to back up each table space. If most of the data in a database is in one table space, the backup is serialized while only the one table space is being backed up. Having data equally distributed between multiple table spaces ensures maximum parallelism throughout the backup.

- **Difference between system managed space (SMS) and database managed space (DMS) table spaces**
  
  When backing up an SMS table space, every page in the table is included in the backup image. When backing up a DMS table space, only extents that are marked as `used` at the beginning of the backup are included in the backup image.

- **What pages are included in an incremental or delta backup**
  
  An incremental backup includes only pages that were modified since the last `full` backup. Every page must have its page header scanned to determine whether it belongs in the backup. All large object (LOB), long varchar, and long vargraphic (LONG) pages are included in the backup image, because they do not have page headers. A delta backup includes all of the pages that were modified since the last backup of any type. The delta backup also includes all LOB and LONG pages.
When performing incremental backups, DB2 keeps track of whether table spaces have been modified since the last backup. If a table space has not been modified since the last backup, the scanning of that table space can be skipped. However, if even one page has been modified in a table space, every page in the table space must be scanned to find the modified page. This can result in the `db2med` EDU being idle for a long time, and can potentially cause storage managers to think that the backup is hung and to close down the connection to the `db2vend` process. This results in a failed backup. Make sure to set idle timeout values with the storage manager appropriately.

Incremental and delta backups are designed to be smaller than full database backups, not faster. If the incremental or delta backup can skip the scanning of some table spaces, the backup might be faster than a full database backup. If all table spaces must be scanned, the extra scanning of the page headers to determine whether a page belongs in the backup image might cause the backup to take longer than a full database backup.

**Backup performance considerations**

Consider several items to maximize the performance of database backups:

- **Buffer size and number of buffers**
  DB2 automatically picks an optimized value for the number and size of the backup buffers. The buffers are allocated from the utility heap, which is controlled by the `util_heap_sz` database configuration setting. The minimum buffer size is 8 (4 KB) pages and the maximum buffer size is 16384 (4 KB) pages. The ideal size of all combined buffers is 90% of the utility heap for offline backups, and 50% of available space in the utility heap for online backups. The number of buffers is dependent on the number of `db2bm` and `db2med` EDUs selected for the backup. Default values for the number of buffers and the size of buffers can be overwritten by specifying 'WITH num-buffers BUFFERS' and 'BUFFER buffer-size' in the `BACKUP DATABASE` command. The size of the backup buffer is +1 of a multiple of the extent size. This plus one page is to accommodate the object header that precedes each block of user data in the backup image.

- **Parallelism**
  At the start of the backup, each `db2bm` EDU is assigned to back up an individual table space. The table spaces are backed up in order of decreasing size. The size is determined by multiplying page size by the number of used pages. When a `db2bm` has completed one table space, it processes the next largest table space in the list. If there are no remaining table spaces, the `db2bm` is idle until the backup is completed. The number of `db2bm` EDUs created is based on the number of processors (CPUs) available, and can be manually adjusted using the `PARALLELISM` parameter of the `BACKUP DATABASE` command.

  At the start of the backup, one or more `db2med` EDUs are started. If the database is being backed up to disk, the number of `db2med` EDUs is based on the number of output directories that are listed in the `TO` clause of the `BACKUP DATABASE` command. If the database is being backed up to a storage manager, the number of `db2med` EDUs is based on the 'OPEN <N> SESSIONS' option of the `BACKUP DATABASE` command. Each `db2med` EDU writes to its own output device.

- **Prefetching**
  When a table space has a single container, the reads for the table space are performed directly by the `db2bm` EDU. When a table space has multiple containers, the read operations are performed by the `db2pfchr` (prefetcher) EDU instead of directly by the `db2bm`. This can enable faster reads because parallel I/Os can be issued across multiple containers at the same time.
For a single container table space, prefetching can still be used by setting the registry variable `DB2_PARALLEL_IO=*` and altering the table space to make prefetch size be a multiple of extent size. The instance must be restarted for the `DB2_PARALLEL_IO` registry variable to take effect.

When using prefetching for backups, ensure that the `NUM_IOSERVERS` database configuration parameter is greater than or equal to the number of `db2bm` EDUs created.

- **Order that table spaces are backed up**
  
The table spaces are backed up in order of decreasing size. The size is determined by multiplying the page size of the table space by the number of used pages in the table space. User or system temporary table spaces are never included in the backup image.

- **Fragmentation of free extents**
  
  When backing up a DMS table space, the `db2bm` EDU reads blocks of contiguous used extents, up to the size of the backup buffer. If there are free extents below the high water mark, the block of data must be broken up into smaller blocks.

  Each block of data must be read in *exclusive* mode. Concurrent disk reads from online activity can conflict with the reads of the backup. It is critical to minimize the number of reads to maximize backup performance. This can be done both by reducing the number of free extents below the high water mark, and by performing an online backup during times of lighter user activity.

  Viewing the output of the `db2pd -tablespaces` command is an easy way to look for free space fragmentation:

  ```
  db2pd -db sample -tablespace 2
  
  Database Member 0 -- Database SAMPLE -- Active -- Up 0 days 00:03:37 --
  Date 2016-08-25-12.07.39.893090

  Tablespace 2 Configuration:
  Address  Type  Content  PageSz  ExtentSz  <…> RSE  Name
  0x00007FECA5A64080  DMS  Large  4096  32  <…> Yes  USERSPACE1

  Tablespace 2 Statistics:
  Address  TotalPgs  UsablePgs  UsedPgs  <…> FreePgs  HWM  <…>
  0x00007FECA5A64080  1155264  1155232  805376  <…> 349856  1150496  <…>
  
  In this example, used pages (`UsedPgs`) is 805376 and the high water mark (`HWM`) is 1150496. 1150496 - 805376 indicates 345120 free pages below the HWM. Dividing the number of free pages below the HWM by the HWM (345120 / 1150496) indicates that 30% of the pages below the HWM are free.

  If reclaimable storage is enabled for the table space, consider using the `ALTER TABLESPACE` command with the `REDUCE` or `LOWER HIGH WATER MARK` parameter to consolidate all of the used pages at the beginning of the table space, and potentially reduce the size of the table space. There are two ways to check whether reclaimable storage is enabled:

  - In DB2 V10.5 and later, run the `db2pd -db <database> -tablespaces` command and look at the `RSE` column to determine if reclaimable storage is enabled for a particular table space. The possible values are Yes and No.
  - Issue the following command (one line):
    ```
    db2 "select varchar(tbsp_name, 20) as tbsp_name, reclaimable_space_enabled from table(MON_GET_TABLESPACE('',-2)) as t"
    ```
A value of 0 indicates that reclaimable storage is not enabled. A value of 1 indicates that reclaimable storage is enabled.

Reclaimable storage is enabled for all table spaces that are created in DB2 V9.7 and later.

- **Compression**
  
The `COMPRESS` parameter of the `BACKUP DATABASE` command causes the data in the backup buffer to be compressed before being written to the storage device. If a custom compression library is not specified, DB2 uses a default compression algorithm. This default compression algorithm is CPU-intensive, and can cause the backup to slow down, especially if the system is already at or near its CPU capacity. The compress function is called by the `db2bm` EDU each time it is ready to place a backup buffer on the full queue. When using compression, make sure to monitor for a shortage of backup buffers. See the `DB2_BAR_STATS` section for monitoring details.

An alternative to using the CPU-intensive compress parameter is to enable compression at the table level. See the following website for a discussion about the various types of compression available: http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/t0055284.html

Any of these compression options can significantly reduce the size of the table, and therefore reduce the size of the backup image.

- **Throttling**
  
The `BACKUP DATABASE` command supports throttling of the backup utility by using the `UTIL_IMPACT_PRIORITY` parameter. The backup runs unthrottled by default. When throttling is enabled, the EDUs involved in the backup periodically invoke the sleep system call. The sleep call yields the CPU to the next thread that is ready to run. The additional sleep calls cause the throttled backup to run longer than an unthrottled backup. If throttling is being used to reduce the run queue at the operating system level, consider adding more CPU to the system.

**Monitoring backup performance with the DB2_BAR_STATS registry variable**

Starting with DB2 V10.1 FixPack 2, every successful backup or restore operation prints statistics in the `db2diag.log` file. This feature also exists in DB2 V9.7, but must be enabled by using the `DB2_BAR_STATS` registry variable.

The meanings of the various columns are as follows:

- **BM.** The `db2bm` EDU ID.
- **Total.** Length of time that each EDU existed.
- **I/O.** Time that was spent performing read or write I/O.
- **MsgQ.** Time that was spent waiting for an I/O buffer.
- **WaitQ.** Time that was spent waiting for a state machine control message.
- **Buffers.** Number of I/O buffers that were processed.
- **Kbytes.** Quantity of data that was processed.
- **MC.** The `db2med` EDU ID.

If the backup was performed using the `COMPRESS` option, two additional columns are present:

- **Compr.** Time that was spent performing the compression operation.
- **Compr Bytes.** Quantity of uncompressed data that was compressed.
Example 1

The following example depicts a scenario in which the db2bm EDUs spend more than optimal time waiting on a free buffer:

<table>
<thead>
<tr>
<th>BM#</th>
<th>Total</th>
<th>I/O</th>
<th>MsgQ</th>
<th>WaitQ</th>
<th>Buffers</th>
<th>kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>11.66</td>
<td>8.17</td>
<td>3.02</td>
<td>0.37</td>
<td>31</td>
<td>460544</td>
</tr>
<tr>
<td>001</td>
<td>11.66</td>
<td>8.59</td>
<td>2.22</td>
<td>0.78</td>
<td>29</td>
<td>460544</td>
</tr>
<tr>
<td>002</td>
<td>11.66</td>
<td>8.21</td>
<td>3.36</td>
<td>0.00</td>
<td>35</td>
<td>567440</td>
</tr>
<tr>
<td>003</td>
<td>11.66</td>
<td>7.94</td>
<td>3.29</td>
<td>0.37</td>
<td>29</td>
<td>460544</td>
</tr>
<tr>
<td>004</td>
<td>11.66</td>
<td>7.00</td>
<td>3.30</td>
<td>1.27</td>
<td>29</td>
<td>460544</td>
</tr>
<tr>
<td>005</td>
<td>11.66</td>
<td>7.52</td>
<td>3.29</td>
<td>0.77</td>
<td>29</td>
<td>460544</td>
</tr>
<tr>
<td>006</td>
<td>11.66</td>
<td>7.12</td>
<td>3.55</td>
<td>0.91</td>
<td>29</td>
<td>460544</td>
</tr>
<tr>
<td>007</td>
<td>11.66</td>
<td>7.57</td>
<td>3.02</td>
<td>0.98</td>
<td>29</td>
<td>460544</td>
</tr>
<tr>
<td>008</td>
<td>11.66</td>
<td>7.78</td>
<td>3.20</td>
<td>0.60</td>
<td>29</td>
<td>461120</td>
</tr>
<tr>
<td>009</td>
<td>11.66</td>
<td>7.18</td>
<td>3.65</td>
<td>0.77</td>
<td>29</td>
<td>460928</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MC#</th>
<th>Total</th>
<th>I/O</th>
<th>MsgQ</th>
<th>WaitQ</th>
<th>Buffers</th>
<th>kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>13.65</td>
<td>7.68</td>
<td>3.99</td>
<td>0.00</td>
<td>299</td>
<td>4867268</td>
</tr>
<tr>
<td>000</td>
<td>13.65</td>
<td>7.68</td>
<td>3.99</td>
<td>0.00</td>
<td>299</td>
<td>4867268</td>
</tr>
</tbody>
</table>

In this example, the db2bm EDUs spend 27% (31.95/116.66) of the time waiting on a free buffer (MsgQ).

There are two potential options to address this issue:

- Increase the number of buffers. The downside of this option is that it reduces the size of the buffers, and potentially result in more, smaller I/O operations.

- Allocate more db2med EDUs, either by specifying more devices in the TO clause of the BACKUP DATABASE command, or by increasing the SESSIONS value.
Example 2

The following example depicts a scenario in which the db2bm EDUs spend more than optimal time compressing data:

Parallelism = 5
Number of buffers = 10
Buffer size = 16781312 (4097 4kB pages)

<table>
<thead>
<tr>
<th>BM#</th>
<th>Total I/O</th>
<th>Compr</th>
<th>MsgQ</th>
<th>WaitQ</th>
<th>Buffers</th>
<th>kBytes</th>
<th>Compr kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>10.71</td>
<td>3.90</td>
<td>6.76</td>
<td>0.00</td>
<td>0.00</td>
<td>6</td>
<td>567440</td>
</tr>
<tr>
<td>001</td>
<td>10.71</td>
<td>3.62</td>
<td>5.35</td>
<td>0.00</td>
<td>1.69</td>
<td>3</td>
<td>460544</td>
</tr>
<tr>
<td>002</td>
<td>10.71</td>
<td>3.60</td>
<td>5.36</td>
<td>0.00</td>
<td>1.68</td>
<td>3</td>
<td>461504</td>
</tr>
<tr>
<td>003</td>
<td>10.71</td>
<td>3.80</td>
<td>5.34</td>
<td>0.00</td>
<td>1.51</td>
<td>3</td>
<td>460544</td>
</tr>
<tr>
<td>004</td>
<td>10.71</td>
<td>3.54</td>
<td>5.44</td>
<td>0.00</td>
<td>1.68</td>
<td>3</td>
<td>460544</td>
</tr>
<tr>
<td>TOT</td>
<td>53.58</td>
<td>18.47</td>
<td>28.28</td>
<td>0.00</td>
<td>6.58</td>
<td>18</td>
<td>2410576</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MC#</th>
<th>Total I/O</th>
<th>MsgQ</th>
<th>WaitQ</th>
<th>Buffers</th>
<th>kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>10.79</td>
<td>0.37</td>
<td>10.36</td>
<td>0.00</td>
<td>19</td>
</tr>
<tr>
<td>TOT</td>
<td>10.79</td>
<td>0.37</td>
<td>10.36</td>
<td>0.00</td>
<td>19</td>
</tr>
</tbody>
</table>

In this example, the db2bm EDUs are spending 53% (28.28/53.58) of the time compressing data. One alternative is to use table-level compression so that the tables are stored in compressed format in the database. This option eliminates the need to do compression during every backup.

Example 3

The following example depicts a scenario in which the db2bm EDUs spend too much time in WaitQ:

Parallelism = 3
Number of buffers = 3
Buffer size = 16781312 (4097 4kB pages)

<table>
<thead>
<tr>
<th>BM#</th>
<th>Total I/O</th>
<th>MsgQ</th>
<th>WaitQ</th>
<th>Buffers</th>
<th>kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>12.13</td>
<td>0.05</td>
<td>0.00</td>
<td>12.03</td>
<td>3</td>
</tr>
<tr>
<td>001</td>
<td>12.12</td>
<td>11.59</td>
<td>0.10</td>
<td>0.03</td>
<td>175</td>
</tr>
<tr>
<td>002</td>
<td>12.12</td>
<td>0.58</td>
<td>0.22</td>
<td>11.29</td>
<td>7</td>
</tr>
<tr>
<td>TOT</td>
<td>36.38</td>
<td>12.23</td>
<td>0.32</td>
<td>23.36</td>
<td>185</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MC#</th>
<th>Total I/O</th>
<th>MsgQ</th>
<th>WaitQ</th>
<th>Buffers</th>
<th>kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>13.45</td>
<td>4.47</td>
<td>7.67</td>
<td>0.00</td>
<td>186</td>
</tr>
<tr>
<td>TOT</td>
<td>13.45</td>
<td>4.47</td>
<td>7.67</td>
<td>0.00</td>
<td>186</td>
</tr>
</tbody>
</table>

In this example, 2 out of 3 db2bm EDUs spend almost all of the time in WaitQ. A high WaitQ value indicates that the db2bm EDU was idle for most of the backup. The number of Buffers processed is also heavily skewed to one db2bm EDU, which indicates that there was one table space in this database that was significantly larger than the rest. Redistributing data so that the table spaces are equally sized improves backup performance.
Related information

For more information, see the following web page:

IBM DB2 V10.5 for Linux, UNIX, and Windows documentation
This document was created or updated on October 21, 2016.

Send us your comments in one of the following ways:

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

This document is available online at http://www.ibm.com/redbooks/abstracts/tips1344.html.

**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.

DB2®
IBM®
Redbooks®
Redbooks (logo)®

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.