Process your DB2 for i indexes in parallel

Kent Milligan

August 19, 2010
(First published January 09, 2003)

With IBM® DB2® for i, parallel processing is useful for more than just queries. Batch processing, data loads, and index builds can all occur much faster if you use multiple processors to do that work. *[2010 Aug 19: The author updated this article with content based on enhancements in DB2 for i Version 7.1. --Ed.]*

Introduction

When people think of parallel processing in a database engine, they immediately think of improved *query* response time by having multiple processors working on a single query. What people tend to overlook is that parallel processing can speed up *index* processing, too, and thus improve the overall performance of your database server. With IBM DB2 for i, it is possible to use CPUs for index processing. Specifically DB2 for i can use multiple processors when creating indexes and when maintaining indexes as the underlying data changes. The ability to create and maintain indexes in parallel applies to both the traditional binary radix and encoded vector index structures.

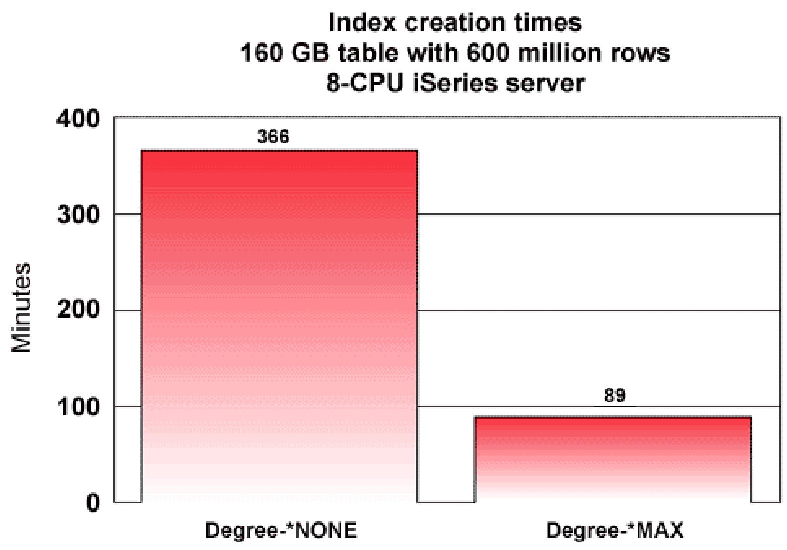
The DB2 for i parallel processing is only available on IBM i systems where the DB2 Symmetric MultiProcessing (DB2 SMP) licensed feature has been purchased, installed, and activated, as described in the Enabling parallel processing section of this article.

Parallel index creation

DB2 for i can use parallel processing for the creation of both SQL indexes and keyed-logical files (that is, the CRTLF command). Parallel index processing is accomplished by logically breaking the underlying table into multiple segments of data. Then each process builds the index key values for the assigned table segment. The work performed by each parallel process is then merged together to complete the final index structure.

Figure 1 shows that using additional CPU resources can substantially reduce the amount of time it takes to create an index. This ability becomes very important in a very large database environment or in a recovery scenario where recreating the indexes needs to occur as quickly as possible.

Figure 1. Index build times are greatly improved with parallelism enabled



Parallel index maintenance

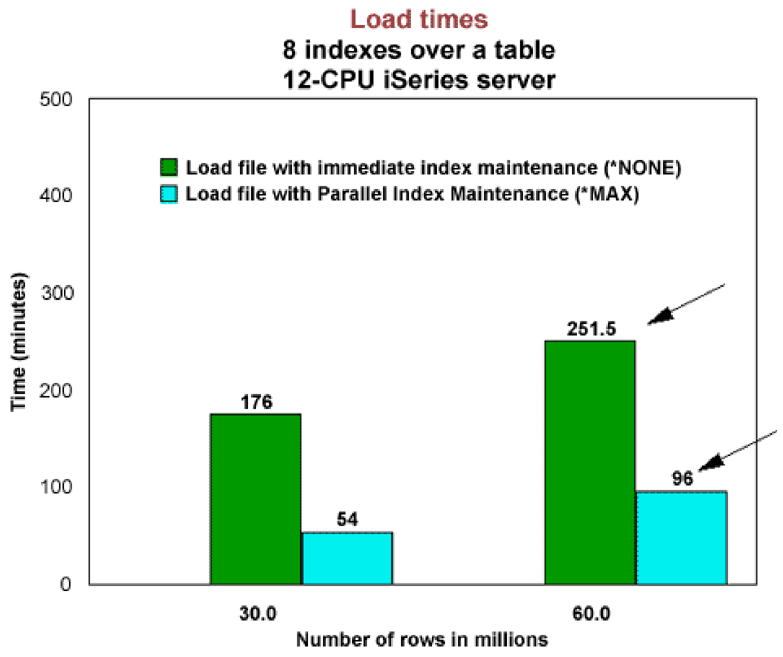
Index maintenance is the task of changing the index key to reflect changes (usually caused by `insert`, `update`, or `delete`) to the associated database row. If an order number for a customer is changed or if a new order for a customer is added to the database, all indexes that include the order number column as a key field must be updated. The maintenance (or update) of the indexes factors into the overall transaction time associated with adding or changing an order.

If more than one index is affected by a database row change, the maintenance of those indexes occurs serially, by default, one index at a time. First, the order number change is propagated to IndexA. After the change to IndexA is completed, the order number is propagated to IndexB, and so on.

Parallel index maintenance involves maintaining multiple indexes concurrently. The order number changes for a customer can be propagated to IndexA and IndexB simultaneously. This concurrent index maintenance processing reduces the total amount of time it takes to get an order number change through the database. Parallel index maintenance trades resources for time, increasing the I/O velocity of a given application or transaction.

Figure 2 shows the time savings that can be achieved by maintaining multiple indexes in parallel. In Figure 2, new rows are being loaded into a table with multiple indexes, and parallel index maintenance improves the load time by two to three times.

Figure 2. Load time improves when index maintenance occurs in parallel



Assuming that the DB2 SMP feature has been installed and activated, DB2 for i uses parallel index maintenance only when the application is:

- Performing blocked inserts (or writes) of eight or more rows.
- Inserting into a table that has two or more indexes defined over it.
- Inserting into a table that does not reuse deleted rows (such as REUSEDLT(*NO)). This requirement can be temporarily overridden.

Blocked inserts are commonly found in batch update and data warehouse load processes, so parallel index maintenance has the biggest performance impact in these environments.

Tip: If a load process is adding more than 20% new rows to a table, it's usually recommended to drop the indexes before the load and then use parallel processing to rebuild the indexes after the load process has completed.

Overriding reuse-deleted-rows requirement

When a table is reusing deleted rows, DB2 cannot perform row-level blocking at the database engine level, because DB2 must extract the individual rows out of the blocked insert so that each individual insertion can be placed in a deleted row slot. As a result, the lack of blocked inserts at the database engine level prevents parallel index maintenance from being used.

Parallel index maintenance can be enabled by turning off the reuse-deleted-rows behavior for a table. However, disabling reuse of deleted rows can have a negative impact because users might need to frequently reorganize their DB2 tables in order to reclaim the storage space for deleted rows. In fact, DB2 for i creates all SQL tables with the reuse-deleted-rows feature to virtually eliminate the requirement to reorganize tables.

As a result, DB2 offers a new Override Database File (OVRDBF) command option in the 7.1 release to provide the capability to temporarily enable DB2 row-level blocking and parallel index maintenance for tables defined with the Reuse Deleted attribute set to *YES. By specifying a parameter value of REUSEDLT(*NO) on the Override command, the job requesting the override will have the table treated by DB2 as if it was created with REUSEDLT(*NO). Batch jobs can be modified to invoke this override in order to speed up their performance because REUSEDLT(*NO) will enable the batch process to benefit from both DB2 row-level blocking and parallel index maintenance (assuming DB2 SMP has been installed). All other jobs that access the same table will honor the permanent reuse-deleted-rows setting of *YES.

For performance benefits of this new override option on the IBM i 5.4 and 6.1 releases, you can purchase an asset from the IBM Lab Services organization (see Related topics for contact information).

Enabling parallel processing

After you install the DB2 SMP feature, there are several different ways to activate parallel processing so that parallel index builds and maintenance can be performed.

At a system level

By default, the amount of parallel processing is controlled at a system-wide level using the **QQRYDEGREE** system value. If that system value is set to a value other than `*NONE` or `*IO`, DB2 for i will use parallel processing when maintaining and building indexes. The `CHGSYSVAL` (Change System Value) commands can be used to change the setting of the QQRYDEGREE system value. Following are the values that you can specify to enable parallel index processing:

- `*MAX` - The DB2 for i engine can use all of the CPU processors and all of the memory available to the job (connection) during parallel index maintenance and build.
- `*OPTIMIZE` - A *good neighbor* setting. The DB2 engine uses its job share of the CPU processors and memory.
- `*IO` - The query optimizer can choose to use any degree of parallelism to perform I/O processing.
- `*NONE` - This is the default value. No parallel processing is used for index processing.

At a job or connection level

If you want to restrict parallel processing to an individual job or connection, use one of the following interfaces to override the system value setting:

- `CHGQRYA` (Change Query Attributes) CL system command
- `QAQQINI` options file
- `SET CURRENT DEGREE` SQL statement

CHGQRYA interface

The DEGREE parameter on the CHGQRYA CL command accepts the same values as the system value. In addition, the DEGREE parameter also supports the value of `*NBRTASKS` *n* where n can be any numeric value between 2 and 9999. This value sets the maximum number of CPU processors that the DB2 engine can use. It's not recommended to use this parallel degree setting.

If the CHGQRYA command in Listing 1 is executed in a job, the parallel degree is set to `*MAX` only for that job, regardless of the system value.

Listing 1. CHGQRYA command

```
CHGQRYA DEGREE(*MAX)
```

This command can be issued multiple times within a job to turn parallelism on and off.

QAQQINI interface

The `QAQQINI` file also lets you specify parallel processing for an individual job or connection. Like `.INI` files found on your computer that are used to store the configuration settings that control the behavior of PC tools and applications (colors, window size, and so on) , the `QAQQINI` file can be used to influence the behavior of the database engine, including the parallel processing behavior. These `QAQQINI` configuration settings can be saved and applied dynamically across multiple database requests.

The fact that the `QAQQINI` file is just a normal database table means you can dynamically change the values of different attributes as your environment changes. The table is also very flexible, because normal database interfaces can be used to change the values of a `QAQQINI` attribute. Listing 2 shows how to use the `QAQQINI` file to set the parallel processing degree to `*OPTIMIZE`.

Listing 2. QAQQINI command

```
UPDATE MyLib.QAQQINI SET QQVAL = '*OPTIMIZE' WHERE
   QQPARM='PARALLEL_DEGREE'
```

The `PARALLEL_DEGREE` attribute supports a superset of the parallelism values supported by the QQRYDEGREE system value and the CHGQRYA command. In addition to the parallel degree values supported by those interfaces, the `PARALLEL_DEGREE` attribute supports the following values:

- `*MAX` *p* - The optimizer performs its normal parallel degree calculation for the `*MAX` setting. After that parallel degree calculation is finished, the parallel degree value is increased or decreased using the percentage value supplied with *p*. The value of *p* can be from 1 to 200. Any value less than 100 decreases the parallel degree used on the database request, and any value greater than 100 can increase the parallelism degree. For example, a value of `*MAX`*50* would cut the parallel degree in half, and a value of `*MAX`*200* would double the parallel degree setting.
- `*OPTIMIZE` *p* - This value causes the optimizer to behave in a similar way to the `*MAX`*p* setting. The difference is that the percentage is applied to the parallel degree calculation for the `*OPTIMIZE` setting.

There is no limit on the number of times that a `QAQQINI` attribute value can be changed during a job or connection, so it's easy to use parallel processing for some DB2 operations and to turn it off for other DB2 operations using the interface. See Related topics for more information about creating and using the `QAQQINI` file.

SET CURRENT DEGREE interface

The SET CURRENT DEGREE statement provides a direct SQL interface for changing the parallel degree value. Listing 3 shows that the SQL statement syntax is quite simple.

Listing 3. SET CURRENT DEGREE value

```
SET CURRENT DEGREE = MAX
```

The parallel degree values SQL interface supports are slightly different than the other interfaces. The following is a list of the supported parallel degree values for the `SET CURRENT DEGREE` interface:

- `MAX` - Equivalent to the *MAX parallel degree setting
- `ANY` - Equivalent to the *OPTIMIZE parallel degree setting
- `IO` - Equivalent to the *IO parallel degree setting
- `n` - Equivalent to the *NBRTASKS *n* setting, when the value of *n* is greater than or equal to 2
- `NONE` or `1` - Equivalent to the *NONE parallel degree setting

Conclusion

This article has shows the value of the parallel processing provided by the DB2 SMP feature for non-query workloads. To enable parallel index processing, activate parallel processing using one of the interfaces described in this article before you build indexes, load data, or perform batch processing. By doing so, you can more fully utilize CPU resources on a multi-CPU server and speed up performance.

Related topics

- Go to the IBM Lab Services organization page to purchase the override option for the IBM i 5.4 or 6.1 releases. It is possible to arrange a no-charge trial of the override option.
- Check out IBM DB2 for i on developerWorks for more information.