Parallel INSERT … SELECT SQL Server 2016

Loading large amounts of data from one table to another is a common task in many applications. Over the years, there have been several techniques to improve the performance of the data loading operations. SQL Server 2014 allowed parallelism for SELECT … INTO operations. However, the users needed more flexibility, in terms of placement of the target table, existing data in the target table, etc., which are not possible with the SELECT … INTO statement. Loading data into an existing table (with or without existing data) through an INSERT … SELECT statement has been a serial operation until SQL Server 2016.

SQL Server 2016, under certain conditions, allows an INSERT … SELECT statement to operate in parallel, thereby significantly reducing the data loading time for these applications. A hidden gem!

Figure 1 illustrates loading time with and without parallelism. The test was performed on an 8-core machine (Figure 3 shows the degree of parallelism achieved), on a table with 50 million rows. Your mileage will vary.

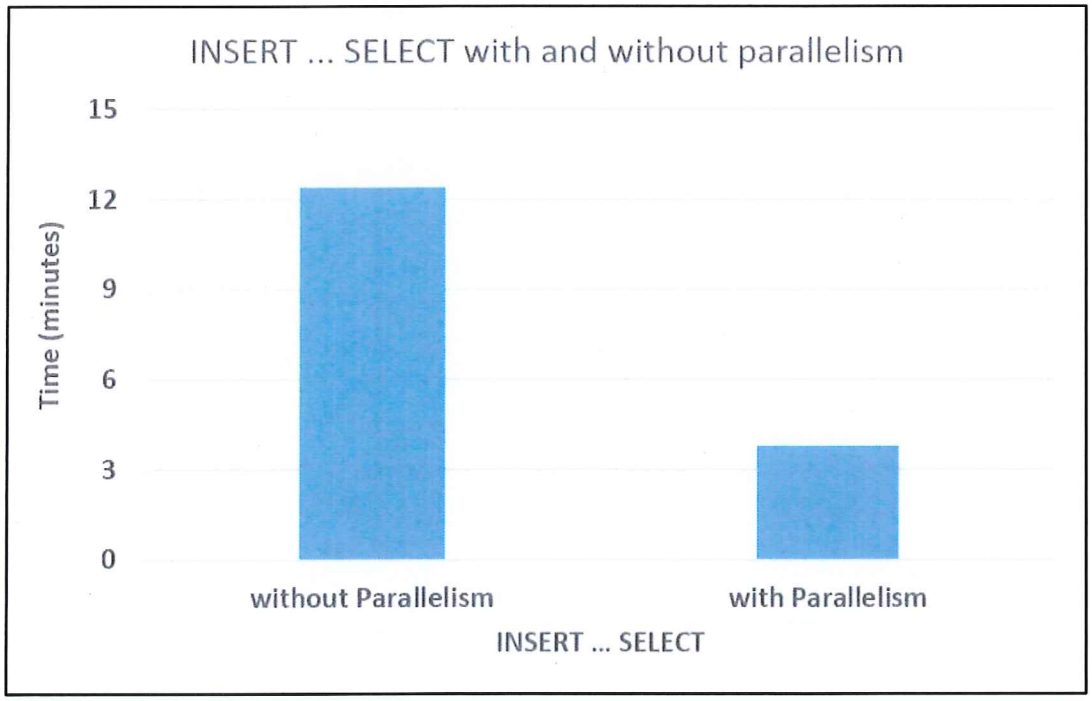


Figure 1: INSERT … SELECT with and without parallelism (SQL Server 2016)

Important to Know

Two important criteria must be met to allow parallel execution of an INSERT … SELECT statement.

1. The database compatibility level must be 130. Execute **"SELECT name, compatibility_level FROM sys.databases"** to determine the compability level of your database, and if it is not 130, execute **"ALTER DATABASE <MyDB> SET COMPATIBILITY_LEVEL = 130"** to set it to 130. Changing the compatibility level of a database influences some behaviour changes. You should test and ensure that your overall application works well with the new compatibility level.
2. Must use the **TABLOCK** hint with the INSERT … SELECT statement. For example: INSERT INTO table_1 **WITH (TABLOCK)** SELECT * FROM table_2.

There are a few restrictions under which parallel insert is disabled, even when the above requirements are met. We will cover the restrictions, and work arounds in the next sections.

How to know you are getting parallelism

The simplest way to check for parallelism is the execution plan. Figure 2 shows an execution plan for an INSERT … SELECT statement without parallelism.

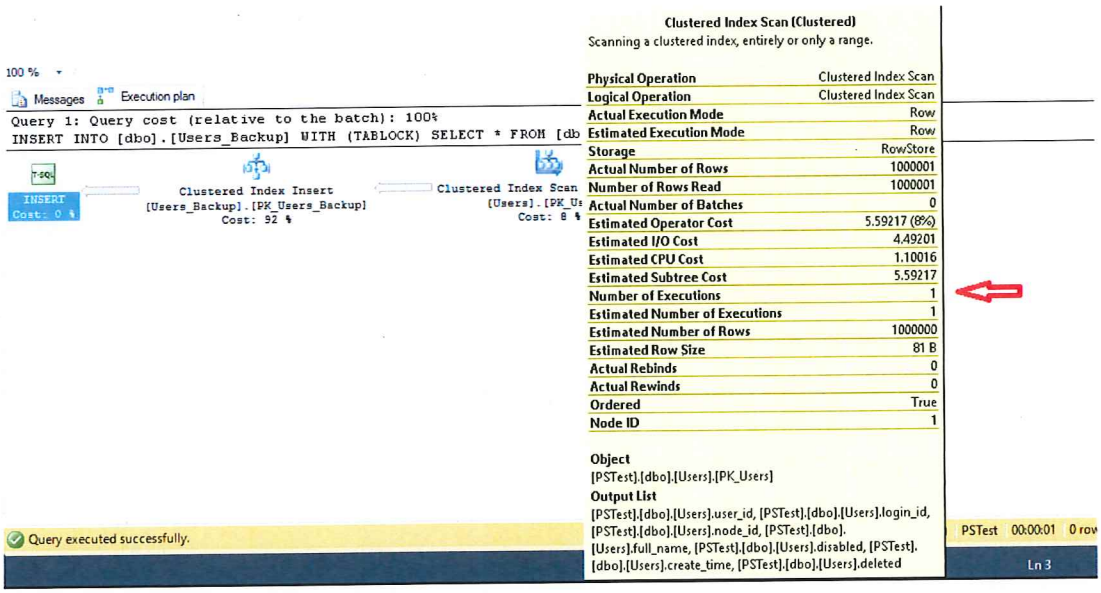
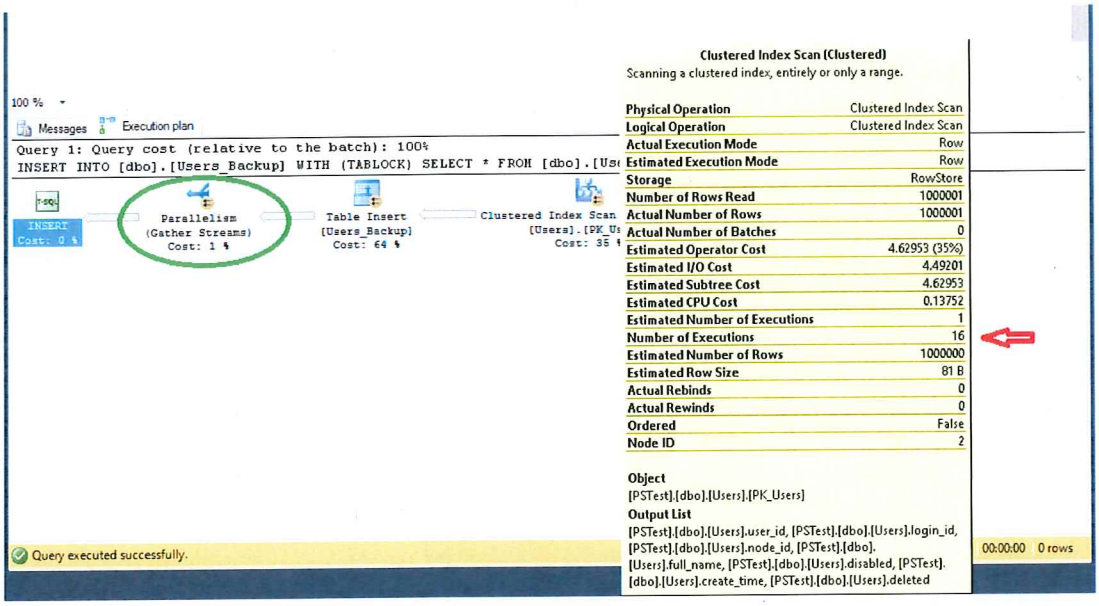


Figure 2: INSERT … SELECT execution plan without parallelism

Under appropriate conditions, the same statement can use parallelism, as shown in Figure 3.



Real World Parallel INSERT...SELECT: What else you need to know!

Here are the considerations and tips to keep in mind when using parallel INSERT...SELECT in the real world.

Have Additional Indexes? Watch out!

It is important to note that the presence of a clustered index or any additional non-clustered indexes on the target table will disable the parallel INSERT behaviour.

Watch out when IDENTITY or SEQUENCE is present!

It is quite common to find IDENTITY columns being used as the target table for INSERT...SELECT statements. In those cases, the identity column is typically used to provide a surrogate key. However, IDENTITY will disable parallel INSERT.

References

- https://technet.microsoft.com/en-us/library/dd425070%28v=sql.100%29.aspx?f=255&MSPPError=-2147217396
- https://msdn.microsoft.com/en-us/library/bb510411(v=sql.120).aspx
- https://msdn.microsoft.com/en-us/library/bb510680.aspx