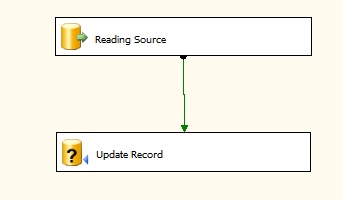
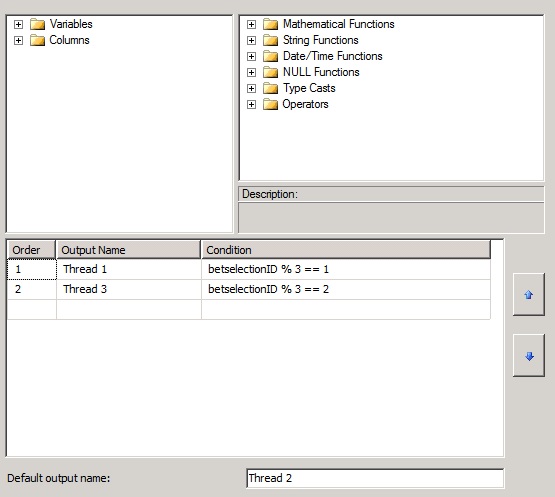
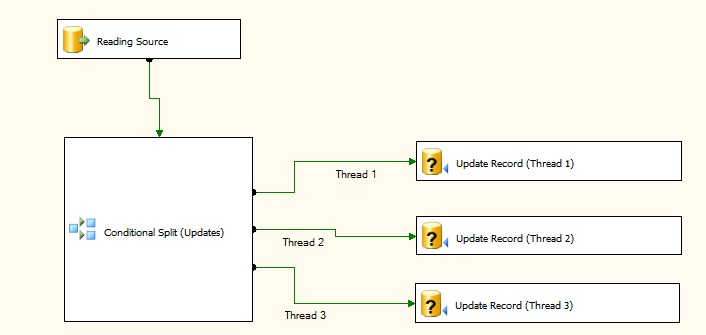
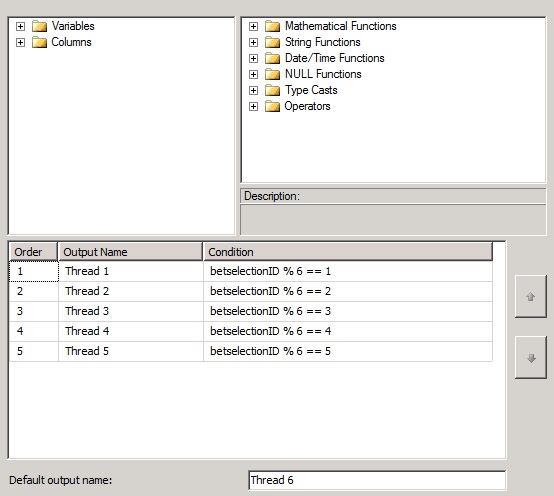
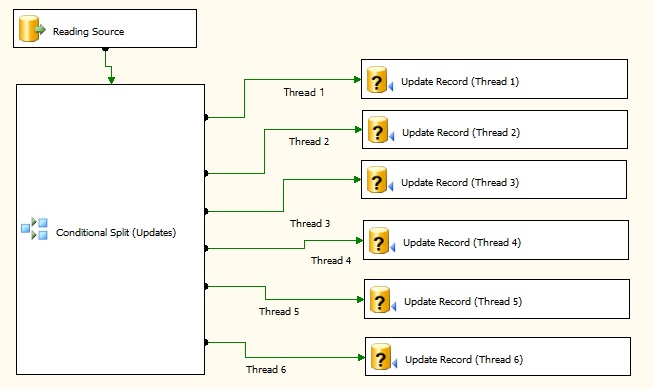
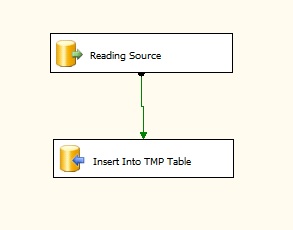
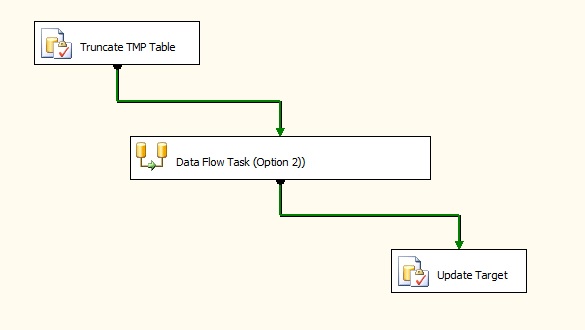
**SSIS PERFORMANCE TECHNIQUES**

**Flat File Connection:**

* Set Fast Parse to TRUE when using Flat File Source.
  + By default SSIS validates any numeric or date columns but with Fast Parse set to TRUE this step will be passed.
  + Right click the Flat File Source or Data Conversion Transformation. Choose Show Advanced Editor. Click the Input and output Properties Tab. Choose the column you want to set the Fast Parse property to TRUE.
* By default columns are set to a STRING data type. To correct this go to the Advanced tab of the connection manager to manually change it. NOTE the “SUGGEST TYPES” button does not always give the desired results.
* **Improving SSIS Update Performance**
* **By** [**Brian Ellul**](http://www.sqlservercentral.com/Authors/Articles/Brian_Ellul/1082043/)**, 2013/10/31**
* I had been entrusted to improve several SSIS packages that form part of a data warehouse ETL load. The area I was to focus upon was the Update phase. Snce the system was already in production, with 100s of packages, I had to offer different solutions weighing in the changes involved vs the expected improved performamce.
* The two options I will highlight below can be applied to any type of package that performs updates, whether to dimension or fact tables.
* **Hardware**
* Although the underlying hardware plays a major role in database and package tuning, in this case I'll obviously be using the same hardware in evaluating the different options. The hardware I'm using is quite basic and it's being used as a development environment.
* I made sure that no one was doing anything on the servers while I took the readings to get a non-distorted picture of the performance improvement.
* **Current System**
* The current system is based on an OLE DB Command updating a table. I have removed all the other processing from the package to make it easier to follow and just left the Reading Source component (which is an OLE DB Source) and a destination component which is an OLE DB Command. Inside the OLE DB Command, I have a simple UPDATE statement where I'm updating all table columns using either the Surrogate Key or the table's unique columns.
* This setup is giving me approximately 99K executions/min (as seen from the Activity Monitor)
* 
* **Option 1 (Multiple concurrent (3) Updates)**
* Option 1 is based on the principle of increasing the UPDATE threads that are sent to the SQL Server. In this example I've bumped up the UPDATE threads to three. This new setup gives me appoximately 255K executions/min, much better than the previous option.
* This is simply achieved by splitting the incoming Source into three threads using a conditional Split (below). I'm applying the [SSIS modulus](http://technet.microsoft.com/en-us/library/ms141815.aspx) fuction to the Source ID and using the Integer remainder to split into multiple threads.
* The image below is showing the Conditional Split component as setup for 3 threads.
* 
* The image below is displaying how data flow will look once the additional update components have been created. This change is fairly simple and it's more a matter of copy & paste of components.
* 
* http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)
* **Option 2 (Multiple concurrent (6) Updates)**
* Option 2 is also based on the principle of increasing the UPDATE threads that are sent to the SQL Server. In this example I've bumped up the UPDATE threads to six. This new setup gives me appoximately 400K executions/min, better than the previous option and much better than the original setup.
* As in option 2, this is simply achieved by spliting the incoming Source into six threads using a conditional Split (below).
* The image below is showing the Conditional Split component as setup for 6 threads.
* 
* http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)
* The image below is a screenshot of the new data flow utilizing 6 update threads.
* 
* http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)
* **Option 3 (INSERT then UPDATE)**
* This option will involve the most development effort to implement. Here, the records requiring an Update (in our example, all records to keep it simple) will first be Inserted into a Temp Table using the fast Bulk Insert. This table will have the same structure as the final destination table, however without any column constraints & foreign keys. At this stage we need the fastest possible Inserts and a heap table is our best bet for this task. Once all records have been inserted into the temp table, an Update statement is performed on the destination table reading from the temp table linking to either the surrogate key or unique columns.
* 
* For example, the statement will look like;
* Update <Destination Table>
* From <Temp Table>
* Where <Destination Table>. Surogate Key = < Temp Table> . Surrogate Key
* The image below is showing the required changes. The first SQL statement is being used to truncate the temp table before executing the data flow while the Update Target SQL taskis used to execute the UPDATE statement.
* 
* http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)
* **Results**
* The table below details the results I obtained on performing the mentioned changes to the package. The source has been limited to 1 million rows, enough to get a fairly accurate representation of the real world performance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Option Name** | **Processing Time** | **% Faster** | **Changes required** |
| 1 | Current Setup | 12 mins | 0% | None |
| 2 | Option 1 (Multiple (3) concurrent Updates) | 4.54 mins | 62.17% | Moderate |
| 3 | Option 1 (Multiple (6) concurrent Updates) | 3.15 mins | 73.75% | Moderate + |
| 4 | Option 2 (INSERT then UPDATE) | 2.25 mins | 81% | Most |

* Option 1 is the current setup and is the slowest. No development changes are needed if the performamce is satisfactory or the number of updates on the destination table is very low.
* Options 2 & 3 do involve some development changes. Both are faster than Option 1, and both options will take approximately the same effort, (option 3 slightly more). No major changes are required and therefore less UAT time will be spent to confirm and check the changes.
* *Note: An observation regarding options 2 & 3: increasing the number of threads will not continue increasing the executions/min simply because eventually the server's IO limit will be reached and no further increase will be possible. It must be emphasised that this option demostrates that a simple modification will aid in performance, however the ultimate improvement will be achieved through option 4.*
* Option 4 is the fastest however it will involve the most changes to the package.
* **Conclusion**
* As can be noted above, more than one solution can exist to speed up these kind of packages, the ultimate deciding factors being performamce vs developement effort. From a business point of view, option 4 may be the way-to-go but this involves the most changes and might not be feasible to implement due to time/HR constraints. On the other hand, if this package is part of an ETL process and the acceptable processing time window is being breached, then option 4 may be the way to go since it provides the best timings.

# Practices to increase the performance of SSIS Packages

By [Bhavin Barot](http://www.codetails.com/author/bbc172038) · [Leave a Comment](http://www.codetails.com/bbc172038/increasing-the-performance-of-ssis-package-best-practices/20121107#comments)

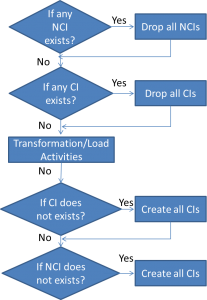
SSIS – **SQL Server Integration Services** is an improved version of DTS (Data Transformation Services) and is widely being used for ETL (Extract, Transform and Load) operations.  It provides an excellent set of tasks and features that can be used in all data related activities such as extraction and applying any transformations with great performance.  And above all! This package,  widely being used by Database Developers and Database Administrators, comes free with SQL Server installation so if you are using SQL Server you do not need to apply/buy any other licenses for SSIS.

This article describes best practices to enhance and boost performance of a SSIS package.  You may find several articles on the Internet to boost the performance, but this gives you a holistic picture and represents my views

## ****Handling Indexes while data transformation****

When working with any data in RDBMS, indexing plays a vital role in boosting search operations.  Clustered and Non-Clustered Indexes both may exist in source and/or destination data. When amount of data is small, it doesn’t matter if indexes are managed run-time or indexes are managed separately.  However, with a large amount of data involved while transformation, unmanaged indexes hit the performance greatly.  You can see a drastic difference in the performance when transforming large amount of un-indexed data.

To handle this, first make the target table a heap by dropping all indexes created on it. Then, transfer the data to heap. At the end of the transformation, create an index on the target table. Here is a flowchart to do this activity.

[](http://www.codetails.com/bbc172038/increasing-the-performance-of-ssis-package-best-practices/20121107/ssis_bestpractise1)

***Legend:*** NCI = Non Clustered Indexes; CI = Clustered Indexes

## ****Avoid using select \* in data flow task****

Data Flow task is one of the most commonly used task of Integration Service. In Data Flow task,  SSIS uses a buffer to do Transfer and Transformation task.  The size of the buffer is dependent on several factors. One of them is the **Estimated Row Size**. The Estimated Row Size is determined by summing up the maximum size of all columns in a row.

*Estimated Row Size = Sum of Maximum size of ALL columns in a Row*

So more the number of columns at source, less number of rows in the buffer at destination.

Hence, it is recommended to select a minimum number of columns at source which are required at destination.  So if we use the query “select \* ” on source connection it will reduce the buffer size. Even if we need all the columns from source, it is better to write down the name of the columns rather than using asterisk (\*).  When we use asterisk (\*), it takes another round for the source to gather the metadata about the columns.

## ****Avoid using Table or View name in Variables****

SSIS has provided a great feature of using variables and expressions within Script Task, Connection Managers and alike. It is recommended to avoid using Table name or View name in a variable for a source/destination connection manager. This is because, SSIS internally queries using “select \*” clause on the variable used.   And as described earlier, this can reduce your performance

## ****Destination Connection settings****

While setting up OLE DB Destination connection, consider following settings.

Data Access Mode : This setting provides the ‘fast load’ option which internally uses a BULK INSERT statement for uploading data into the destination table instead of a simple INSERT statements (for each single row) as in the case for other options. So unless you have a reason for changing it, don’t change this default value of fast load. If you select the ‘fast load’ option, there are also a couple of other settings which you can use as discussed below.

Keep Identity : By default this setting is unchecked which means the destination table (if it has an identity column) will create identity values on its own. If you check this setting, the dataflow engine will ensure that the source identity values are preserved and same value is inserted into the destination table.

Keep Nulls : Again by default this setting is unchecked which means default value will be inserted (if the default constraint is defined on the target column) during insert into the destination table if NULL value is coming from the source for that particular column. If you check this option then default constraint on the destination table’s column will be ignored and preserved NULL of the source column will be inserted into the destination.

Table Lock : By default this setting is checked and the recommendation is to let it be checked unless the same table is being used by some other process at same time. It specifies a table lock will be acquired on the destination table instead of acquiring multiple row level locks, which could turn into lock escalation problems.

Check Constraints : Again by default this setting is checked and recommendation is to un-check it if you are sure that the incoming data is not going to violate constraints of the destination table. This setting specifies that the dataflow pipeline engine will validate the incoming data against the constraints of target table. If you un-check this option it will improve the performance of the data load.

## ****Setting up Rows Per Batch and Maximum Insert Commit Size Settings****

**Rows per batch :** The default value for this setting is -1 which specifies all incoming rows will be treated as a single batch. You can change this default behavior and break all incoming rows into multiple batches. The allowed value is only positive integer which specifies the maximum number of rows in a batch.

**Maximum insert commit size :** The default value for this setting is ’2147483647′ (largest value for 4 byte integer type) which specifies all incoming rows will be committed once on successful completion. You can specify a positive value for this setting to indicate that commit will be done for those number of records. You might be wondering, changing the default value for this setting will put overhead on the dataflow engine to commit several times. Yes that is true, but at the same time it will release the pressure on the transaction log and tempdb to grow tremendously specifically during high volume data transfers.

The above two settings are very important to understand to improve the performance of tempdb and the transaction log. For example if you leave ‘Max insert commit size’ to its default, the transaction log and tempdb will keep on growing during the extraction process and if you are transferring a high volume of data the tempdb will soon run out of memory as a result of this your extraction will fail. So it is recommended to set these values to an optimum value based on your environment.

## ****Avoid Asynchronous Transformation wherever possible****

SSIS runtime executes every task other than data flow task in the defined sequence.  Whenever the SSIS runtime engine encounters a data flow task, it hands over the execution of the data flow task to data flow pipeline engine.

The data flow pipeline engine breaks the execution of a data flow task into one more execution tree(s) and may execute two or more execution trees in parallel to achieve high performance.

Synchronous transformations get a record, process it and pass it to the other transformation or destination in the sequence. The processing of a record is not dependent on the other incoming rows.

Whereas the asynchronous transformation requires addition buffers for its output and does not utilize the incoming input buffers.  It also waits for all incoming rows to arrive for processing, that’s the reason the asynchronous transformation performs slower and must be avoided wherever possible. For example, instead of using Sort Transformation you can get sorted results from the source itself by using ORDER BY clause.

## ****Use SSIS performance Counter where required****

SSIS has great event logging mechanism that helps analyse the performance of the package and its components.  SSIS has also introduced System Performance counter counters to monitor the performance of your SSIS runtime and data flow pipeline engines.

For example,

* **SSIS Package Instance counter** indicates the number of SSIS packages running on the system;
* **Rows read and Rows written counters** indicate the total number of rows coming from the source and total number of rows provided to destination;
* **Buffers in use and Buffer memory counters** indicate the total number buffers created and amount of memory used by them;
* **Buffer spooling** is a very important counter and tells about number of buffers (which are not currently in use) written to the disk when physical memory runs low;
* **BLOB bytes read, BLOB bytes written and BLOB files in use counters** give detail about the BLOB data transfer and tells about number of BLOB bytes read, written and total number of files that the data flow engine currently is using for spooling BLOB data etc.