

# Integration Services: A Strategy for Performance

SQL Server Technical Article

Writers: Elizabeth Vitt, Intellimentum, and Hitachi Consulting

Contributors: Donald Farmer, Microsoft Corporation; Ashvini Sharma, Microsoft Corporation; Stacia Misner, Hitachi Consulting

Published: October 2005

Updated: October 2005

Applies To: Microsoft® SQL Server™ 2005

Summary: This document describes how to successfully use a performance strategy for optimizing Microsoft® SQL Server™ 2005 Integration Services data integration solutions.

## Copyright

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only**. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT**.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

 2005 Microsoft Corporation. All rights reserved.

Microsoft and SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

## Table of Contents

[Introduction 1](#_Toc132440366)

[Modeling Performance 1](#_Toc132440367)

[Operations 2](#_Toc132440368)

[Defining Operations 2](#_Toc132440369)

[Sharpening Operations 3](#_Toc132440370)

[Volume 4](#_Toc132440371)

[Defining Volume 4](#_Toc132440372)

[Sharpening Volume 4](#_Toc132440373)

[Application 5](#_Toc132440374)

[Understanding Application Usage 5](#_Toc132440375)

[Assessing Alternatives 6](#_Toc132440376)

[Location 7](#_Toc132440377)

[Defining Location Choices 7](#_Toc132440378)

[Assessing Location Impact 7](#_Toc132440379)

[Conclusion 8](#_Toc132440380)

[About the Authors 10](#_Toc132440381)

[Elizabeth Vitt, Intellimentum 10](#_Toc132440382)

[Hitachi Consulting 10](#_Toc132440383)

#### Introduction

Performance is a make or break success factor for data integration solutions. Performance not only determines how effectively you can satisfy your current data integration requirements, but also influences how well your solutions can scale and how quickly they can be adapted to changing business needs. For example, these can include increasing data volumes, shrinking batch windows, expanding data complexity, and emerging data quality standards.

Although performance plays a critical role in every project, ironically it is most often addressed at the end of the development lifecycle. At this point, however, the design is nearly complete, deadlines are looming, customers are becoming increasingly anxious, and you are running out of options to tweak the design.

Optimizing an existing design certainly has great value, but you may find that your tuning options are limited when you have made particular design decisions. A certain amount of after-the-fact performance tuning is inevitable. However, you can mitigate the amount of last minute tuning by incorporating an ongoing performance strategy into your design methodology. Such a strategy can help you make informed design decisions throughout the development lifecycle, help you deliver highly tuned solutions in a more predictable and timely manner, and save precious hours during project crunch time.

Microsoft® SQL Server™ 2005 Integration Services (SSIS), a full-featured data integration engine and development environment for building high performance data solutions, introduces a data integration performance strategy that helps you design better design SSIS solutions by addressing four performance elements. These consist of operations, volume, application, and location.

#### Modeling Performance

Given the number of variables that apply to a specific data integration solution, a useful performance strategy needs to be broad enough to adapt to a variety of situations, but also specific enough to offer you actionable information. This is precisely the goal of the performance strategy for SSIS: to provide guidance that helps you assess a data load situation, identify the primary factors that impact data integration performance, and maximize the impact of specific tuning techniques. The performance strategy’s four key performance factors are defined as follows:

* **Operations** – To identify potential tuning opportunities, first break down your integration solution into specific tasks or units of work and then fully define all data operations that are part of each task. Some of these operations may be easily identifiable, but some of the best performance tuning opportunities may be less obvious.
* **Volume** – With the operations fully defined, identify the anticipated data volumes at each stage of the process and think about how the volumes impact the performance of data operations. This is also a great opportunity to consider how data volumes are expected to change over time.
* **Application** – Given the operations and volume specifications, select the SQL Server application and/or technology that best fits the job. There is always more than one way to accomplish a task; the key is to identify which application provides you the right amount of functionality with the least amount of overhead.
* **Location** – The final factor is location, which refers to the run-time environment for the data integration operations. Location is last because the parameters for the other performance factors will influence how you optimally configure the run-time environment.

A simple mnemonic, OVAL, will help you to remember these four components of a successful performance strategy.

To demonstrate the benefits of this approach in the sections that follow, you will use these components to optimize a simple data integration solution. In this scenario, you are using SQL Server Integration Services to aggregate a 5,000,000-record flat file and load the resulting records into a destination SQL Server RDBMS table.

#### Operations

Operations is the key factor of the performance strategy. It highlights the need to understand all the data operations that are part of your solution so that you can identify the opportunities to sharpen and tune. This means going beyond the obvious tasks at hand to identify the complete sequence of operations from the data source through each data integration operation to the destination.

##### Defining Operations

To demonstrate how operations works, review the operations displayed in Table 1. This table describes each operational step involved when using SSIS to load and aggregate flat file data into a SQL Server destination table.

Table 1 – Operations to load and aggregate a flat file into a SQL Server table

|  |  |
| --- | --- |
| Operation Step | Performance Considerations |
| 1. Read data from the text file | In this step, SSIS establishes a connection to the source file and reads the records. How fast the records can be read from the source depends to a large degree on factors external to SSIS, such as disk speed, I/O contention, network speed, and machine resources.  Even though these factors may be external to your SSIS application, it is important to recognize how they impact overall performance. If there are performance bottlenecks at this step, you might consider an alternative approach. For example, to reduce I/O contention, break apart the source file so that you can use sequential read operations. |
| 1. Load data from the text file into the SSIS Data Pipeline | In this step, SSIS parses the source file and loads into SSIS memory structures called buffers. During design time, SSIS establishes rules for parsing the file by collecting metadata such as data types, sort order, and parsing routine. At execution time, the file is read and parsed according to these rules. The more precise the rules, the more efficient the process. |
| 1. Aggregate data | In this step, SSIS aggregates the data that has been read into the SSIS Data Pipeline. To perform the aggregation, SSIS creates a cache that you can configure to improve performance and maximize how SSIS utilizes memory resources. |
| 1. Open a transaction on SQL Server | After the aggregation is complete, a series of steps occur as SSIS loads the aggregated data into a table in the destination SQL Server database.  At this point, all the established best practices for optimizing bulk loading in SQL Server come into play, such as configuring the Database Recovery Model, enabling Fast Load, using Table Locks, and configuring the Commit Size. |
| 1. Pass data to the SQL Server interface |
| 1. Load the data into each SQL Server destination table |
| 1. Commit the transaction |

##### Sharpening Operations

For each operation in this sequence, you can use the performance considerations to guide your search for opportunities to sharpen and optimize the overall process. For example, in operation 2, when SSIS loads data from the text file into the Data Pipeline, all fields are read as string data types, even if they contain integers. If your goal is to load this file into a SQL Server destination table with named, typed columns, then data conversions have to occur at some point during the data load process.

To reduce the size of the data set and improve the performance of the overall data load, you should configure data types early in the sequence of operations. In the example described in Table 1, the optimal time to configure data types is during operation 2, when the file is loaded into the SSIS Data Pipeline. If you do not configure data types at this point, data passes through all steps in its original string state until it is converted upon load into SQL Server.

To further tune file parsing, you might also take advantage of the SSIS fast parsing mode in operation 2. This mode uses a fast parsing routine for simple data loads in which the data contains common date formats, does not require locale-specific parsing, does not have special characters in currency columns, and does not have hexadecimal characters.

Data type optimizations do not just apply to file data sources. Setting data types early in the SSIS data load process, as appropriate, can provide performance benefits for all data sources.

This is just one example of how you can sharpen your data load operations. You should apply this approach to each step in the process so that you can identify alternative design approaches for each performance consideration.

#### Volume

Volume is the second critical, significant factor influencing performance. Higher volumes have a higher impact on system resources such as memory and CPU. The more you understand how volume impacts your data operations, the more you can tune resource utilization to produce more efficient operations.

##### Defining Volume

The SSIS Data Pipeline Engine uses a buffer-oriented architecture that efficiently loads and manipulates data sets in memory. The value of this in-memory processing is that data does not have to be physically copied and staged at each step of the data load. Instead, the Data Pipeline Engine manipulates data by using buffers as it moves data from source to destination.

To optimize the Pipeline, your goal is to pass as many records as possible through the fewest number of buffers, especially when you are dealing with large data volumes. To accomplish this goal, you must first analyze the length (number of rows) and width (number of columns) of your data. The width is roughly determined by the combined size of the data types of the columns, plus a fixed memory overhead that SSIS allocates. The smaller the data types, the more efficient the memory usage.

##### Sharpening Volume

When you sharpen data volumes, you should first assess all the columns that are part of your data integration operation. Many times it is tempting to just load in an entire source file even though you may only need half of the columns. Unnecessary columns take up machine resources and should be eliminated where possible.

Another factor to consider is whether your solution requires incremental or full data loads. From a volume perspective, full data loads tend to be costly. If you can replace full data loads with incremental data loads you will greatly reduce the volume of your data load operations. This is especially so when volumes are likely to increase over the lifecycle of an application.

Within SSIS, you can sharpen volume in three primary ways: shrinking the size of your data set, maximizing throughout, and defining parallel operations.

1. **Shrinking Data Set** **Size** - To illustrate the concept of shrinking data sets, consider the following example. Assume that the 5,000,000-record source text file has 100 columns of numbers between 1 and 999. If you leave the default string data type for these columns, each column will need 50 bytes per column per record. This means that the calculated size of each row will be approximately 5,000 bytes, plus some additional memory overhead that SSIS requires. So, with a volume of 5,000,000 records and 5,000 bytes per row, the total size of the file will be approximately 23 GB. Consider the difference in the size of the data set if you define the columns more accurately as 2-byte integers. This will reduce the size per row to 200 bytes and the total size of the file to 954 MB, a savings of approximately 22 GB over the string data type. With this small change in data type size, you now have far less data to load into memory. Note that this example is included to illustrate the relative difference of data types and the savings involved with shrinking the size of the data set. It is not intended to be a hard and fast sizing guideline.
2. **Maximizing Throughput** - In addition to configuring data types, you can also maximize throughput by configuring SSIS buffer settings such as DefaultMaxBufferSize and DefaultMaxBufferRows. These settings control how many buffers are created during a data load operation as well as how many records pass through each buffer. The specifics of tuning these settings are beyond the boundaries of this white paper.
3. **Defining Parallel Operations** - The third way to sharpen volume is to consider how you can parallelize operations to maximize efficiency. SSIS supports parallel processing of packages, tasks, and transformations. When you design parallel processes, you need to consider how parallelism applies to all data load operations. For example, if you need to load and aggregate multiple files into SQL Server, you might consider performing one or more of the following operations in parallel: file reading, aggregation, and destination table loading. Although SSIS supports the parallelism of all these operations, you still have to be mindful of your machine resources and the level of parallelism that they can support. You might build a highly parallel SSIS package and then later realize that your machine does not have enough CPU and memory to take full advantage of the parallelism.

Although the example of loading 5,000,000 source records is somewhat simple, these same principles directly apply to larger and more complex data loads where you have multiple data sources, require complex data transformations, and must load to multiple destinations.

#### Application

With a defined set of data integration operations and an assessment of the required data volumes, addressing the Application factor helps you determine which data integration application can best meet those requirements.

##### Understanding Application Usage

You will often have a choice of using different applications to solve data integration problems. The key is identifying which application provides you with the right amount of functionality and with the least amount of overheard.

For example, you may have situations where using the SQL Server Transact-SQL BULK INSERT statement or bcp utility to load text files into SQL Server provides the right amount of data load functionality with a minimal amount of overhead, as compared to SSIS. These situations likely occur when you have small data sets and simple data integration requirements where the overhead of starting an SSIS operation offsets the performance gain of using bcp or BULK INSERT.

To guide your decision, list the major functionality components of each application so that you can assess how they relate to your primary requirements. For example, the following usage guidelines can help you decide between BULK INSERT/bcp or SSIS.

* **BULK INSERT / bcp Usage Guidelines** - Use BULK INSERT / bcp when your scenario has the following characteristics:
* A single data source and it is a file.
* A single destination and it is SQL Server.
* No data transformation requirements such as direct load from source to destination.
* No workflow management. Note that workflow management is not really applicable to BULK INSERT functionality, because its sole job is to load data into SQL Server.
* **SSIS Usage Guidelines** - Use SSIS when your scenario has the following characteristics:
* Multiple heterogeneous data sources and destinations.
* Data transformation requirements: aggregations, lookups, and new columns.
* Incorporation of other management tasks such as Email and FTP.
* Workflow management to control the order of many tasks and transformations.

##### Assessing Alternatives

With a list of usage guidelines, you can better assess how each application alternative meets your data requirements.

In the file data load example, your requirements are not only to load 5,000,000 records into SQL Server, but also to aggregate those records. With these requirements in mind, consider the following alternative approaches to performing this task. Note that these are just two possible approaches.

* **Alternative 1 - BULK INSERT Solution** - You could certainly use BULK INSERT to load the data from the source file into SQL Server. However, you would also need to include additional operations to aggregate the data before loading it into the final destination. One way to accomplish this would be to use BULK INSERT to load the data into a staging table, use Transact-SQL to aggregate the data from the staging table, and then use Transact-SQL to load into the destination table.
* **Alternative 2 - SSIS Solution**- SSIS offers a one-stop shop where you can both aggregate and load data to the destination table. In addition, within SSIS you can add workflow to control the order of operations and even set up execution logging to record the progress of operations and also the error handling to capture any invalid records.

When you compare the two previously listed alternatives, Alternative 2, the SSIS solution, eliminates the need for a staging table and encapsulates all data load logic and workflow into a single SSIS package. To achieve the same functionality with Alternative 1, the BULK INSERT solution, you would not only have to use Transact-SQL to aggregate data, but you might also have to consider using it to encapsulate all the data logic in a stored procedure. So, based on this comparison, the SSIS solution proves to be an excellent choice for the aggregate-and-load example, thus providing the right amount of functionality with the ability to centralize and manage workflow.

#### Location

Finally, you need to identify the location where data operations execute. As you load data from source to destination, there may be situations in which you have flexibility as to where you execute specific operations. Likely, your decision will be based on a series of tradeoffs that you will have to evaluate, depending on your specific data load needs.

##### Defining Location Choices

Typically, you have one of three location choices for execution: (1) data source server, (2) data destination server, or (3) dedicated extract, transform, and load (ETL) application server. In reality, your choices may be limited by licensing restrictions. However, by considering the impact of each location on your application requirements, you will determine the best fit for your execution needs. You might also find a need to re-evaluate your licensing agreements.

##### Assessing Location Impact

To illustrate the impact of location, consider where the source file data is located relative to the location where the data load operation executes and also the location of the SQL Server destination:

1. **Data Destination Server** – When your destination is SQL Server, executing SSIS operations on the destination data server provides significant performance advantages. The biggest advantage in this situation is the ability to use the SQL Server Destination component in an SSIS package. This component optimizes in-memory data loading and results in an 8 to 15 percent performance improvement over the standard OLE DB destination component. Keep in mind that the SQL Server destination component does not support data type conversions. As a result, you will definitely have to complete conversion operations in an SSIS step prior to loading data into the destination. This is so that the data types in memory directly line up with the SQL Server data types.

You will also have to consider how SSIS competes with SQL Server for machine resources on the destination server. To reduce contention, you can configure resource usage by either SQL Server or SSIS. For example, one change you can make to SSIS is to lower the maximum number of concurrent SSIS threads to avoid CPU bottlenecks. Overall, if your data load operations require you to find the fastest way to load data into SQL Server, then running SSIS operations on the data destination server will provide substantial performance benefits with SQL Server in-memory data loading.

1. **Data Source Server** – When you execute SSIS operations on the data source server, you can gain performance benefits by reducing the source data set size before transferring data across the network to load into the destination. For example, if you need to aggregate the data, the number of output records will likely be smaller than the number of input records. In addition, you can reduce the size of a data set by properly configuring data types early in the SSIS process.

To get these benefits, you have to accept a few tradeoffs. If other applications run on the source server, your SSIS routines might compete with these applications for machine resources. If resources are somewhat limited, you will likely offset any performance benefits that you would have received. In addition, if SQL Server is your destination, you will not be able to use the optimized in-memory data loading feature of the SSIS SQL Server Destination component. The SSIS SQL Server Destination is only useful when the SSIS data operation is executed on the destination SQL Server machine. Otherwise, you will have to use the standard OLE DB destination component. However, it cannot take advantage of in-memory data loading.

Considering these tradeoffs, if you can reduce a large volume of records into a much smaller data set, executing the SSIS operations on the source data server may prove to be the most beneficial if you are concerned about network performance bottlenecks.

1. **Dedicated SSIS Server** – Using a dedicated SSIS server is the best solution when you have a variety of heterogeneous data sources and destinations with large volumes and complex transformations. On this dedicated server, SSIS can take full advantage of all machine resources and will not suffer from contention issues with other applications.

For the tradeoffs, once again you will not be able to use the SSIS SQL Server Destination component. In addition, from a network perspective, you will have to transfer data twice: from the source onto the SSIS server and then from the SSIS server into the destination. If your network bandwidth can easily support this, then this option provides the most SSIS flexibility. It also allows for performing complex transformations without adversely affecting other applications or SQL Server.

#### Conclusion

As you design data integration solutions, you can treat performance tuning as an art or even as a science, but you do not want to leave the outcome to chance. To help you take the guesswork out of performance tuning, the performance strategy described in this white paper provides a structured approach to help you build high performing data integration solutions.

By guiding you through an assessment of operations, volume, application, and location, this strategy can help you anticipate and optimize the interaction of these variables in simple as well as complex and large scale data load scenarios.

For more information:

<http://www.microsoft.com/technet/prodtechnol/sql/2005/technologies/ssisvcs.mspx>

#### About the Authors

##### Elizabeth Vitt, Intellimentum

Elizabeth Vitt has ten years of business development, project management, consulting, and training experience in business intelligence. Her industry experience includes BI implementations in retail, manufacturing, and financial services. She has specialized experience as an educator in data warehousing, ETL, and OLAP design and implementation. Ms. Vitt is an author of Microsoft Official Curricula courses for Microsoft Business Intelligence product offerings as well as the MSPress, *Business Intelligence: Making Better Decisions Faster*. In anticipation of the launch of SQL Server 2005, Ms. Vitt has successfully implemented SQL Server 2005 for early adopter customers.

##### Hitachi Consulting

Hitachi Consulting, the global consulting company of Hitachi, Ltd. (NYSE: HIT), is a recognized leader in delivering proven business and IT solutions to Global 2000 companies across many industries. Hitachi Consulting leverages decades of business process, vertical industry, and leading-edge technology experience to understand each company’s unique business needs. From business strategy development through application deployment, Hitachi consultants are committed to helping clients quickly realize measurable business value and achieve sustainable ROI.

Hitachi Consulting is a Microsoft Certified Gold Partner for Business Intelligence and an exclusive provider of curriculum and instructors for the Microsoft SQL Server 2005 Business Intelligence Ascend training program. Hitachi Consulting is also an experienced systems integrator and has successful SQL Server 2005 BI implementations at companies participating in Microsoft’s Technology Adoption Program (TAP).

Hitachi Consulting offers a client-focused, collaborative approach and transfers knowledge throughout each engagement. For more information, visit <http://www.hitachiconsulting.com/>.

Hitachi Consulting – Inspiring your next success®