IBM WebSphere Transformation Extender

Performance Recommendations

Version 8.1

Note
Before using this information, be sure to read the general information in "Notices" on page 39.

October 2006

This edition of this document applies to IBM WebSphere Transformation Extender Version 8.1; and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, e-mail DTX_doc_feedback@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Introduction

IBM WebSphere Transformation Extender offers the flexibility and power to solve integration needs. This flexibility provides a user with a variety of options to develop their transformation processes and the power to accomplish their objectives in the most optimum ways.

This documentation contains general performance tips and recommendations to improve performance. The purpose of this documentation is to present strategies for leveraging the flexibility and power of IBM WebSphere Transformation Extender so that transformation needs can be met while achieving reasonable performance objectives.

Note that not all recommendation will apply to all platforms.

Chapter 2. General performance

This documentation describes what takes place during product execution and presents performance tips and recommendations regardless of the platform on which a specific IBM WebSphere Transformation Extender product is running. This information will provide you with some additional background to assist in defining performance objectives.

Overview

There are several platform-independent topics for which background information, performance tips, and recommendations are addressed:

- ″Performance Factors″
- ″Understanding Command Server Operation″
- ″Understanding Launcher Operation″
- ″Map Performance Tuning Tips″

Performance factors

When analyzing how well an application has performed, there are potentially a large number of measurements that can be used to gauge workload efficiency in terms of the specific resources the application used. The most common resource used in measuring performance is time. Memory, disk space, and other resource usage are often used, too, because of their impact on performance.

Comparisons can be made between workload and the various resources that impact performance. In this documentation, when the word *performance* is used without a qualifier, it refers to workload/time comparisons. This shortcut is used since this general resource usage is the most common. Other types of comparisons will be explicitly noted by using a qualifier with the word *performance*. The phrase, *all types of performance*, will refer to any workload/resource usage comparison, such as memory and disk space comparisons to workload as well as execution-time comparisons.

For every different application execution, there are a wide variety of performance factors that influence all types of performance for the Command Server and Launcher. The significance of each factor varies for every application execution. Most of these factors are dependent on the performance of other factors; few of the factors are completely independent. Accounting for these inter-relationships among performance factors increases the difficulty in tuning the performance of an application.

Although there can be a wide variety of performance factors, the following list shows a subset of the more significant and commonly-encountered performance factors when running applications using IBM WebSphere Transformation Extender maps. They represent the specific potential areas that you can target for analysis when tuning the performance of your applications. They are listed below organized by logical groupings.

- Mapping
  - Type design
  - Map design

- Hardware
  - CPUs (ISA, MHz, #)
  - Memory architecture
  - Physical memory
- Load
  - Memory contention
  - CPU contention
  - I/O contention
- Compiler
  - C/C++ runtime
  - Code generation
- Input/Output
  - File size
  - File/Disk configuration
  - File system
- Map execution
  - Paging
  - Workspace options
  - Burst/Integral
- Operating system
  - Virtual memory
  - Context switching
  - Load time

Many application executions have a specific constraining factor that impacts performance more so than other factors. Reducing the impact of that factor can provide a more substantial performance improvement than committing time and effort in examining other factors. Often, identifying the constraining factor requires additional time and effort. That additional time is balanced by a more significant performance improvement, which might be sufficient to preclude further analysis of those other factors.

Load and performance measurements

When measuring performance, it is important to account for extraneous load. If a system has a significant amount of usage outside the process being measured, the elapsed time for the process is likely to be longer than when the system has less use. Even though the process has much of the same resource needs, such as memory or CPU processing power, in a busy or a quiet environment, heavy competition for those resources can stall execution. For this reason, elapsed time, or wall time as an example, is not the most appropriate measure of performance for a heavily used system.

Performance metrics should attempt to isolate the process being monitored. Better alternatives to elapsed time include user/system time or CPU time. When measuring CPU use, the proportion used by each application should be noted. Lastly, performance metrics should be done on a system with as little extraneous load as possible.

Occasionally, the combined user/system time or the CPU time of an execution will be acceptable, but the elapsed time is too high. In such cases, reducing the impact

of extraneous load should reduce the gap between the two times. The impact results from a scarcity of one or more resources. Whether the scarce resource is CPU processing power or memory, reducing the impact of external load requires increasing the amount of the contested resources available.

Increasing the amount of a specific resource can usually be done in a number of ways.

For example, if a system is at 100% CPU use because of extraneous load, there are a number of options for reducing that load:

- Tune the application. On a system constrained by CPU power, tuning the application for CPU usage will reduce its overall run time.
- Tune other applications. Reducing the CPU needs of the applications making up the extraneous load will free up some computational power. You should tune the focal application before tuning the other applications. For best results, request assistance from your systems administrator.
- Reduce the application load. Reducing the number of applications running will free up CPU power for the focal application. Unfortunately, this is not an option in many cases.
- Obtain additional processors. Additional processors will increase the amount of CPU power available. Assuming that the extraneous load remains constant, this leaves more CPU power available for the focal application. The amount of improvement will depend on the applications running and the underlying operating system.
- Use faster processors. Replacing the existing processors with faster models will increase the amount of CPU power available.

Because every situation presents its own set of performance variables, each situation must be evaluated to determine the best option to use.

Throughput and CPU expectations

The execution of a single map occurs in the context of a single thread. Therefore, additional processors will not improve the execution time of a single transformation. The Command Server executes a simple map in this fashion. Execution time measurements of a single transformation performed with the Command Server will only account for the computational power of a single processor.

The affect additional processors have on the Command Server contrasts with the affect additional processors might have on the Launcher. Depending on how event triggers occur, separate transformations can execute on multiple processors concurrently. Even in this situation, each transformation still runs in a single thread.

In gauging performance, it is important to choose a measure of performance appropriate to the architecture of the mapping process. The performance measurement should agree with the performance goal for the system as a whole. If the system includes a number of maps running in parallel on multi-processor hardware, transactional throughput is probably more important than the execution time of any single transformation.

For example, a given map might execute in 10 seconds. On a 4-processor box, 4 simultaneous executions of that same map might complete in 11 seconds. The

transactional throughput is nearly four times that of the single execution scenario, but each of the 4 simultaneous executions still executed in 10 seconds.

Conversely, if the process involves a batch of data with a single map, map execution time might be the most appropriate primary measurement. Execution time is also quite appropriate when running a series of maps. In such cases, all of the work occurs in a single thread as previously mentioned.

Assuming that CPU processing power is the constraining factor in both cases, improving throughput and execution time require different approaches. Adding additional processors of the same type will not improve map execution time. Improving map execution time requires more computationally powerful (faster) processors. Conversely, additional processors with the same power can improve throughput. More processors of the same power can allow more maps to execute simultaneously.

This same principle applies when comparing map performance across hardware platforms. A hardware configuration might have more processors available, but if an individual processor lacks computational power, the execution time of a single map on that processor will suffer. However, even with weaker processors, this same configuration might provide better throughput. While computational power varies considerably among processors, clock cycle frequency is a loose predictor of relative computational strength.

Understanding Command Server operation

The first step in trying to achieve performance improvements running the Command Server in your environment is to understand Command Server operation.

Execution overview

After you have developed your map, you are ready to run it using the Command Server. During the map execution process, the transformation engine performs several operations. These operations generally fall under the following phases:

- Initialization
- Validation
- Transformation
- Finalization

Initialization

During initialization, the transformation engine loads maps and prepares data for validation. Adapters are loaded and initialized at this time. This phase of execution also includes allocation of memory for the paging subsystem.

Validation

During validation, the transformation engine parses all the input data sources and identifies objects based on type definitions. The information about the objects within each input data source is stored in one or more temporary files called work files or in memory. Each work file locates and sets boundaries for objects within all input data sources.

Transformation

During transformation, the rules contained within the map are executed. If a map is designed to produce output, the result of this execution process is the expected output data. Map execution proceeds through the output cards, executing each contained rule.

Multiple input values or multiple combinations of data values might cause rules to be executed multiple times. If a rule calls a functional map, the rules contained within the functional map would be executed multiple times.

During transformation, both the workspace and the data sources are heavily used. Paging settings affect how much of these two information sources are available at any one time. You also have the option of creating the Burst and Data Audit Logs in this step.

Finalization

During finalization, temporary work files used during transformation are deleted and other system clean-up tasks are performed. Optionally, the **Map Settings** and **Execution Audit** log creation also occur in this step.

About execution time

Execution time can be affected by various factors. When most transformation processes are executed, the transformation itself consumes the bulk of execution times. Complex validation logic in type trees, especially restarts, can increase the proportion of execution time spent in validation.

The map execution process occurs in a single thread. By using a single thread of execution, the overhead of context switching is avoided in heavily-loaded systems. Excess context switching on heavily loaded systems can quickly dominate execution.

Comparing work and data files

Work and data files are the two major sources of information used in the transformation phase of map execution. Data files are the input data specified at execution time. Work files created during validation contain data about the input and output data sources.

The data types defined by the transformation's type tree depend on the internal format of the data files and are ultimately determined by the business process. This internal format drives the validation phase. The storage characteristics of the data files also depend on the business process as well as the underlying operating system.

IBM WebSphere Transformation Extender offers a significant number of options for creating and using work files. Depending on the conditions of the transformation, such as input data file size and processing complexity, some of these options can significantly affect performance. Reusing Work Files, Paging and WorkSpace in Memory present the performance impact of these options.

The size requirements for work files ultimately depend on the number of type instances found in the input data file and generated in the output data file. The number of type instances found in both cases depends greatly on the type

definitions. To save work file space, type instances are only recorded in the workspace if needed later in the transformation process.

Reusing work files

Work files contain metadata about the various type objects found in each input data file. If the data content is static, this metadata can be reused in subsequent runs. Reusing work files allows the transformation engine to bypass the validation step for these sources. Reducing the number of steps results in a shorter overall execution time.

When possible, reusing work files can produce measurable performance gains, especially in input data files requiring complex validation logic. This option is ideal for map processes that make use of static data, such as lookup tables. Work file reuse is specified separately for each input card, either as an execution command or as an option in the Map Designer. Reusing work files with an execution command is done with the W option on the input card override (`-I`) execution command.

To reuse work files, you must generate a work file.

To generate a work file

1. Specify an input card override `-I` execution command for the corresponding input source for a compiled map with the (`W`) reuse option.
2. Execute the map.

   A work file is generated.

Subsequent map executions with the same input override options will reuse the generated work file. As mentioned previously, this option is ideal if the data content is static. If the data content changes, the work file should be regenerated.

To regenerate a work file

1. Remove the old work file.
2. Specify an input card override `-I` execution command for the corresponding input source for a compiled map with the (`W`) reuse option.
3. Execute the map.

   A work file is generated.

A generated work file can be used to determine the amount of workspace required for a given input source. This can be helpful when using workspace options such as `-WM`.

See the *Command Server* documentation and the *Execution Command* documentation for more information about work file reuse.

Paging

In order to minimize the memory footprint during transformation, IBM WebSphere Transformation Extender keeps only a subset of the work and data file information in memory at a given time. The total amount of memory available for storing data used in the transformation process is segmented into equal-sized portions called pages. Each page holds a contiguous section of either an input or a work file.

Paging and I/O usage

Paging controls the amount of data available to the transformation portion of the map execution at any time. If a map rule requires a piece of data that is not in memory, the paging subsystem will fetch the appropriate data from disk. This can impact existing data in memory; if there are no unused pages, an existing page is swapped out of memory.

In situations where a great deal of information from various areas of the input are needed, increasing the number and size of pages available to the system can reduce I/O volume. Increasing the information window allows the map to process more information without loading pages from disk.

Paging portions of data files allow for processing a large transformation in a much smaller amount of memory. Choosing paging settings appropriate to the transformation can improve performance considerably.

The following paging-related activities affect performance:

- "Using the WorkSpace paging setting"
- "Setting information windows" on page 10
- "Knowing the size of the input" on page 10

Using the WorkSpace paging setting

In many maps, paging settings are explicitly specified in the maps themselves during design. If the designer of a map does not explicitly specify page settings, the page settings for the map will default to a page size of `64 KB` and a page count of `8 KB`. Optimal page size and count settings depend ultimately on the amount of workspace needed. Because the workspace requirements are not available at the beginning of execution, the estimated page size and count will not always result in an optimal execution time. In this situation or if the paging settings within a map are sub optimal, altering paging behavior at execution time might improve performance.

The primary method of altering runtime paging behavior during Command Server execution uses the **WorkSpace PageSize** (`-P`) execution command. With the (`-P`) command, the page size and count can be specified for a transformation. The page count determines the total number of pages available to the system for both data and work file usage. Half of the requested pages are allocated to each file usage. Page size determines the total amount of storage available on each page. Together, the two directly control the amount of information available at any given time, as an information window.

To determine the values for the page settings, you must understand how the transformation engine allocates memory.

When the **WorkArea** card setting on a map is set to **Memory**, the transformation engine will allocate memory as required for both data and workspace, with the page size controlled by the **PageSize** setting in the map.

When the **WorkArea** card setting on a map is set to **File**, the transformation engine will allocate memory to track the information written to file for both data and work space, with the page size controlled by the **PageSize** and **PageCount** settings.

As a result of how memory is allocated depending on the **WorkArea** card setting, if the **WorkArea** is set to **File**, and the **PageSize** and **PageCount** map settings are set to a high number, the transformation engine is more likely to exceed memory.

For most data transformations, a **PageSize** map setting of `1024` kilobytes is too large. Data transformations whose **PageSize** map settings are `64`, `128`, `256` or `384` kilobytes should be tested for performance results using sample data that mirrors the size and complexity of the data that will be used in your production environment. Performance will be impacted negatively if you use **PageSize** map settings that are lower or higher than these recommended settings.

When executing maps in the Launcher, the memory used at startup time includes the totals of all the maps included in the **.msl** files deployed, with the addition of the memory used by any cached maps.

Setting information windows

When specifying paging information, it is critical to understand the information requirements of the map. Certain usage patterns require a larger window into the data and the workspace than other usage patterns. If a paging setting does not provide a large enough window, the paging subsystem will generate additional I/O requests to meet the demands of the map.

Conversely, requesting too much memory through paging settings can have detrimental effects as well. If the amount of requested memory exceeds available physical memory, the underlying operating system will begin paging memory to disk. Selecting appropriate paging settings is a balance between the information needs of the map and the amount of available memory.

Setting an appropriate information window especially applies to series-consuming functions such as `EXTRACT()`, `LOOKUP()`, and `CHOOSE()`. Depending on context, series-consuming functions can require access to the entire data file to complete map execution. This contrasts with the much smaller requirements of simple sequential maps.

Knowing the size of the input

One significant factor in determining appropriate paging settings is the amount of data processed by a map. A larger amount of input might require more paging memory to attain optimal execution time. More paging memory results in a larger information window. The degree to which larger input might require a larger information window varies with the design of the map.

Using a test case

The following test scenario was created to demonstrate how the input size might require a larger information window or page setting. Two test cases were created and then executed using the same testing procedures.

- Maps: The same map is used in both tests.
- Inputs: There are two input data files.
  - Input 1 size: 3 MB
  - Input 2 size: 11 MB
- Process:
  - Map performed a number of operations such as a large number of `EXTRACT()` calls that spanned each of the input data files.

  - These requests mandated that a certain percentage of the input data be available at any given time for optimal performance.
- Objective:
  - Determine the page size to get the optimal execution time for each of the input data files.
- Test Case 1 Result:
  - Quantitative measurement of execution time at various page counts and sizes identified an optimal execution time at the following workpage settings for the 3 MB input data file: size 64 K and count 12. This is represented by `-P64:12`, where `-P` is **WorkSpace Paging**, `64` is the page size and `12` is the page count.
- Test Case 2 Result:
  - The same testing procedure found `-P64:53` to produce the optimal execution time for the 11 MB input data file.
- Conclusion:
  - Paging settings depend on input data size just as much as they do on the map rules. An optimal page setting for a given map and data file might not be optimal for the same map with a much larger or smaller data file.

Finding optimal paging settings

Behavior:

The settings for the paging subsystem within IBM WebSphere Transformation Extender can determine whether a transformation process runs bound by I/O bandwidth or by CPU capacity. A typical execution time/page count curve has two easily discernible parts, based on performance characteristics.

In a graph where the vertical coordinate measures execution time and the horizontal coordinate measures page count, at the low page count end of the graph, the execution time falls quickly as the page count increases. Here, the transformation is I/O bound. The data access patterns implicit in the map require more information than what is available in memory. Because of this, time spent driving I/O dominates execution time.

As page count increases, execution time decreases into a nearly flat portion of the graph. At that stage, the transformation is bound by the computational power of the executing processor. Increasing the amount of memory available through paging settings will probably not improve performance.

Tuning procedure:

Finding the combination of page size and page count that optimizes performance is the goal of the following procedure. There are several steps to follow for tuning paging settings. The settings found for this procedure are specific to the behavior of the map and data used. Because of this, the data should represent future workloads as much as possible.

To tune page settings

1. Verify that the execution environment is stable. Execution times must be relatively repeatable. Run a few simple preliminary executions and time them to gauge the reproducibility of the timing procedure. If the resulting execution times are not similar to one another, then the execution environment is not

stable enough for tuning paging settings. Improve reproducibility by reducing extraneous loads from other applications.

2. Select initial values for page size and count. The overall page size and count specify the total amount of memory available for paging data and workspace. This total amount of memory is allocated early in the execution process. A failure to allocate paging space will result in an early termination. Therefore, a memory request for `C` pages of `S` kilobytes should be expected to succeed, given available storage.
   Appropriate values for each setting vary highly on the data access patterns of the map and the design of the type tree. Not only is each setting a factor in execution time, but the overall amount of memory allocated also affects performance. The tuning procedure here will attempt to insulate page size from the effect of changing overall memory
3. Execute the map and data to be tuned. Note the overall processing time and the processing time spent managing I/O during execution. Sub-optimal paging settings often result in an I/O bound transformation process.
4. Execute the map and data again with different paging settings. Vary the page size up or down. The page count should be adjusted so that the combined paging memory remains as constant as possible. For example, if you double the page size in the second run, the page count should be halved. Again, note the overall processing time and the processing time spent managing I/O during execution. The page count must still be small enough for the entire paging space to fit in the memory allocated to IBM WebSphere Transformation Extender by the operating system.
5. Compare the collected times from both executions after the second execution completes.
6. Repeat this procedure with various page sizes to find the optimal page size for the map.
7. After an optimal page size is determined, a similar set of iterations to vary the page count should be performed, keeping the page size fixed.

   If an increase in memory drastically reduces performance, it is likely that the overall amount of memory has started to throttle the operating system's virtual memory system. If this occurs, reduce the page count. The threshold at which throttling can occur depends heavily on the application load across the entire system. If a decrease in page count drastically reduces execution time, then the execution was probably I/O bound. Compare the difference in I/O usage between executions or the time used by the operating system. An increase indicates that the proportion of execution time devoted to handling I/O has increased. In this case, increase the page count.

Recent changes:

Recent changes in the transformation engine altered the paging behavior of IBM WebSphere Transformation Extender from release 6.7 onward. These recent changes simplify the tuning process by reducing the CPU overhead required to manage larger page counts. Previously, many maps had a single page count setting that would minimize execution time. Choosing a page count larger than this setting would require more CPU bandwidth. Choosing a smaller page count setting would require more I/O bandwidth, putting the execution into an I/O bound state.

WorkSpace in Memory

The **WorkSpace in Memory** option can be a viable choice for reducing I/O usage during transformation if enough memory is available. With a smaller input data file, the transformation engine can attempt to process the work file entirely in memory.

Using this option:

- Significantly reduces the number of I/O requests issued across the execution.
- Produces optimal performance benefits when the work files do not exceed available physical memory. For more information about determining work file size, see the Reusing Work Files
- When producing work files that exceed the amount of available physical memory, will cause operating system level-memory management to simulate additional memory with disk storage. This will ultimately degrade performance.

Understanding Launcher operation

The first step in trying to achieve performance improvements running the IBM WebSphere Transformation Extender Launcher is to understand its operation.

Comparisons to the Command Server

Overall, much of the way the Launcher is designed to execute maps is similar to the Command Server. While the execution of each map still occurs in a single thread of execution, the Launcher offers multiple simultaneous transformations, each in its own thread of execution. Triggering logic specified in therLauncher configuration determines the number of maps that can be executed at any one time.

By launching each transformation in its own thread, the Launcher can take advantage of multiple CPU resources concurrently. Using multiple CPU resources can increase the total throughput and transactional volume. While the transactional volume might increase, the execution time of a simple map, measured against how the Command Server would perform, will be unchanged or possibly longer. The improved throughput results from parallel execution of multiple maps and not faster execution of individual transformations.

Operating system configuration

The ability of the Launcher on a given platform to take advantage of multiple processors is dependant on the configuration of the underlying operating system. More information about Launcher configuration can be found in *Configuration Parameters* and *Launcher Settings* in the *Launcher* documentation.

Map performance tuning tips

Many functions available to users for developing IBM WebSphere Transformation Extender maps are very powerful and can perform difficult and complicated tasks. Some functions might sacrifice performance for powerful capabilities. It is important to understand when it is better to use one function over another that can perform the same task but does so differently. Each function has advantages and disadvantages and users have the flexibility to choose the function that best suits their specific transformation needs.

These are suggestions for tuning maps to achieve the best performance results through the various map functions. The functions fall under the following categories:

- ″Data Searches″
- ″Routing″

There are conversion functions (`BASE64TOTEXT`, `TEXTTOBASE64`, `QUOTEDTOTEXT`, and `TEXTTOQUOTED`) that you can use instead of the following adapters that might also improve performance.

- ″Base64 Adapter″
- ″Quoted-Printable Adapter″

Also, the following utilities can be used to help you identify the areas in your maps that might be causing performance degradation and then you can make the appropriate changes to your maps to improve their performance:

- ″Map Profiler (dtxprof)″
- ″Page Setting Assistant for Maps (dtxpage)″

Data searches

There are many ways to develop IBM WebSphere Transformation Extender maps that search through data. These different options provide varying balances between functionality and performance.

The functions available to the map developer to do data searches are:

- EXTRACT()
- LOOKUP()
- SEARCHUP() or SEARCHDOWN()
- CHOOSE() - see POSITIONAL INDEXING in the table below

The following table contains the functions available to the map developer to do data searches.

Function	Use	Advantage	Disadvantage
`EXTRACT()`	When expected results are multiple objects.	One of the most powerful ways of searching through a data file.	Not the most efficient way to locate objects if the expected result set is only a single object.
LOOKUP()	Where expected results are a single object. Scans a data file object-by-object, looking for an object that matches the specified criteria.	More efficient than `Extract()`.	The average amount of time required for search depends proportionally on the number of objects in the data file.

Function	Use	Advantage	Disadvantage
SEARCHUP() or SEARCHDOWN()	When the set of items to be searched is sorted.	Increased performance. Takes advantage of the sorted order of the data file by traversing the data file as a binary tree. Search time is proportional to $\log_2(N)$, where N is the number of data items.	Cannot be used for unsorted data files. The smaller the input data file, the less performance results benefit.
POSITIONAL INDEXING	When used with CHOOSE() and INDEX(), locates objects within a data file by position.	While executing repeated CHOOSE() requests, MIB maintains a cache of position information for the indexed data file. As long as repeated calls to CHOOSE() reference the same type, this cache continues to be used, and indexing is fast.	Doing repeated calls to CHOOSE() that don't reference the same type, and therefore don't use the cache, indexing is not as fast.

Data search usage

As you develop your IBM WebSphere Transformation Extender maps to search through data, you should consider the input format and the requirements for the output. Based upon those considerations, you can make your decision on which search function to use to for optimum performance results. The following table lists search usage scenarios with the recommended function that should give you the best performance results.

Usage Search Function Recommendation

Single output data item is the expected result and the input data cannot be sorted.

LOOKUP()

Multiple output data items are expected result.

EXTRACT()

The set of items to be searched is or can be sorted.

SEARCHUP() or SEARCHDOWN()

Same data type is being referenced (Positional Indexing).

In cases where CHOOSE() consistently references the same type, execution performance can exceed SEARCHUP() and SEARCHDOWN().CHOOSE()

Comparisons between the search functions

This documentation contains quantitative comparisons of the data search functions. It presents results of comparing the LOOKUP() against the SEARCHUP() or SEARCHDOWN() functions.

The primary difference between the functionality of LOOKUP() and SEARCHUP() or SEARCHDOWN() is the ability to use a full logical expression in LOOKUP(). For example, LOOKUP() can be used to find the first item whose value is less than *3*. SEARCHUP() or SEARCHDOWN() can only find an exact match.

Functionality aside, LOOKUP() and SEARCHUP() or SEARCHDOWN() differ in their performance characteristics. A LOOKUP() search operates by scanning the appropriate set of items in sequence, attempting to match the logical condition. On average, a LOOKUP() search takes time proportional to the number of items in the set.

In contrast, SEARCHUP() and SEARCHDOWN() take advantage of sorted input to provide faster response. Searches with SEARCHUP() or SEARCHDOWN() use a binary search index on the items in the data series. This results in execution time proportional to the logarithm of the item count. In other words, doubling the number of items in the data series causes only an incremental increase in the execution time required for that search.

To achieve this response-time benefit, SEARCHUP() and SEARCHDOWN() require sorted input. Additionally, SEARCHUP() and SEARCHDOWN() can only be used to find an exact match. If these conditions are suitable for the map, then using SEARCHUP() or SEARCHDOWN() can provide significant execution-time benefits. These benefits are especially noticeable for larger data series.

The following table presents the example results.

	Input Size (Bytes)			
Map Functions	**16 KB**	**128 KB**	**1 MB**	**2 MB**
SEARCHUP()	0.23	0.46	3.04	5.83
LOOKUP()	0.28	4.84	288.40	1220.27

This table captures execution times in seconds for two maps that both attempt to match each data element from one input file with another. The two maps differ in the function that each uses to match data. The first map uses SEARCHUP() to locate matches quickly. The second map uses LOOKUP().

For small series of data, the difference in execution time is not as noticeable. However, as the input series size grows, the execution time for the **SEARCHUP** map grows much more slowly than the execution time for the **LOOKUP** map.

A search is executed for every data element in the input file. As a result, the overall execution time grows with the data size *and* the search response time. In the case of the **LOOKUP** map, this yields a map execution time that grows exponentially with input size. A doubling of the input size requires an execution time four times as long for this map.

Routing

Routing is the splitting of output to separate files based on the data values. This is a common operation in many IBM WebSphere Transformation Extender mapping procedures. This documentation discusses differences between the PUT() and RUN() functions when used to route data. The behavior of each is broader than simply routing, and neither function is an exact replacement for the other.

The various functions available to the map developer to do data routing are:

- PUT()
- RUN()

For both the PUT() and RUN() functions, the best performance for routing style maps is generally obtained by mapping the largest available group object.

The following table contains the two functions available to the map developer to do data routing.

Function	Use	Advantage	Disadvantage
PUT()	Routes an output item to an adapter.	Overhead is less than using RUN().	I/O and CPU overhead from repeated opening and closing of routing targets can be great for large maps.
RUN()	Allows a map that is currently running to execute another map.	Is versatile. Using a second map to process output allows for a great deal of flexibility in output generation.	Overhead is much greater than using PUT(). I/O and CPU overhead from repeated opening and closing of routing targets can be great for large maps.

Data routing usage

As you develop your IBM WebSphere Transformation Extender maps to route input data and create output, you consider the format of the input and the requirements for the output. Based upon those considerations, you can make your decision on which routing function to use to derive optimum performance results. The following table lists the routing usage with the recommended function that should give you the best performance results.

Usage Routing Function Recommendation

Avoid using additional maps to route data. Instead, try to use PUT() with the appropriate adapter whenever possible.
PUT

Additional processing with a secondary map is required.
RUN

As stated previously, the functionality of PUT() and the functionality of RUN() differ significantly. Because RUN()enables the map to execute a secondary map, there

might be many routing situations where PUT() might not be a feasible choice. In cases where PUT() can be used, usually better performance results.

Comparisons between the routing functions

This documentation contains quantitative comparisons of the data routing functions. It presents results of comparing the RUN() against the PUT() functions.

In this example, two maps that attempt to route input to any one of twenty possible output targets are executed. The two maps differ in their use of RUN() or PUT() to route data items to an output target.

For each map, two scenarios were tested, each processing input of different sizes. Varying the input was intended to indicate relative scalability between the two functions in a routing context.

The following table presents the example results.

	Input Size (Bytes)	
Map Functions	**128 KB**	**4 MB**
PUT()	0.74	13.43
RUN()	2.14	62.22

This table captures execution times in seconds for the two example maps. Even for input that is smaller in size, PUT() executes faster. For input that is larger in size, the overhead of the additional maps in RUN() processing is far more noticeable. Execution of the **PUT** map completes in nearly one-fifth the time of the **RUN** map.

Comparing type sizes

Regardless of whether PUT() or RUN() is used to route data, there is an execution time overhead in each routing attempt. Minimizing the number of routings performed will improve execution time. Careful attention to map and type tree design can reduce the number of routings done.

This example demonstrates the benefit of minimizing routing attempts. Four maps were run with inputs of two different sizes. The maps differ in two primary ways. They use two different functions, PUT() and RUN(), and different type tree designs.

The two different type tree designs are used for the same input sources. In the first type tree design, **N1**, the input card type tree consists of a simple series of items (0:s). In the second type tree design, **N100**, the input card type tree consists of a series of groups of items. The groups themselves are series, (0:100). Therefore, the second type tree has two levels of series where the first, **N1**, has one.

The following table presents the example results.

	Map Functions			
	PUT()		RUN()	
	Input Size (Bytes)			
Type Tree Design	**128 KB**	**4 MB**	**128 KB**	**4 MB**
N1	0.74	13.43	2.14	62.22
N100	0.36	0.75	0.41	1.83

This table captures execution times in seconds for the four example maps. The savings in execution time are significant, especially at larger input sizes. Whether the map uses `RUN()` or `PUT()`, reducing the number of routes can improve execution time.

Simply grouping input together to route larger amounts of data is not always possible in every map design effort. The degree to which map or type tree design can address route reduction depends on the needs of the business process. This example demonstrates the potential for improving execution time by reducing the number of routing attempts.

Map Profiler (dtxprof)

The map profiler is a user-configurable utility that captures and reports map execution statistics.

The resulting output highlights those areas in your maps, such as component and mapping rules, and the functions and types within those rules, with very long processing times, which might be adversely affecting performance. It might show that specific rules and functions are being executed for an unusual number of times. It also might show that type objects are being accessed too many times or that the nesting level of an object is too deep. With this information, you can then make the appropriate changes to your maps to achieve your performance objectives.

For example, you might have output from the map profiler that indicates that the processing time for a `LOOKUP` function in your map is significantly more than the processing time for other functions. If the data is sorted, you might see the processing time decrease by using the `SEARCHUP` function in your rule instead.

See ″Data Search Usage″ .

Using the Map Profiler

The profiler can be configured and started from the Performance interface using the Map Designer, or from the command line using the *dtxprof* utility command.

See the *Map Designer* documentation for information on how to use the map profiler in the Map Designer GUI.

See the *Utility Commands* documentation for information about how to use the map profiler from the command line, outside the Map Designer GUI.

The map profiler is an important tool to assist you in improving the performance of your maps and achieving your performance goals.

Page Setting Assistant for Maps (dtxpage)

The **Page Setting Assistant for Maps** application is used to calculate suggested settings for memory page size and count for work files used in maps. It is started by using the `dtxpage` utility command. The **.mmc** file is a required parameter and is the file name of the map for which the calculation is being performed.

The application will run iterations of the specified map and therefore you should expect that the overall run time will be longer than a typical run.

See the *Utility Commands* documentation for information about how to use the `dtxpage` utility command from the command line.

The **Page Setting Assistant for Maps** application invoked by the `dtxpage` utility command is an important tool to assist you in improving the performance of your maps and achieving your performance goals.

Base64 adapter

It is recommended for improved performance, that instead of using the Base64 adapter in the command line of your maps, you use the following two Base64 conversion functions in map rules:

- `BASE64TOTEXT`
- `TEXTTOBASE64`

  See the *Functions and Expressions* documentation for information about how to use these functions.

Quoted-Printable adapter

For improved performance, instead of using the Quoted-Printable adapter in the command line of your maps, use the following two Quoted-Printable conversion functions in map rules:

- `QUOTEDTOTEXT`
- `TEXTTOQUOTED`

  See the *Functions and Expressions* documentation for information about how to use these functions.

Chapter 3. Mainframe performance

This documentation explains mainframe performance, what takes place during IBM WebSphere Transformation Extender product execution, as well as performance tips and recommendations for the Batch and CICS execution environments. This information will provide some additional background to assist in defining performance objectives.

Overview

General mainframe performance is discussed in the following topic:

- ″Understanding Mainframe Performance″

Performance tips and recommendations for the two execution environments are discussed in the following topics:

- ″Batch Execution Environment″
- ″CICS Execution Environment″

The ″Command Server Troubleshooting″ information provides recommendations and references to identify potential performance areas when executing IBM WebSphere Transformation Extender under either Batch or CICS environments. Troubleshooting information is available for ″Batch Troubleshooting″ and ″CICS Troubleshooting″.

Understanding mainframe performance

An important step in trying to achieve performance improvements running the IBM WebSphere Transformation Extender under Batch and CICS on the z/OS operating environment is to understand the IBM WebSphere Transformation Extender execution characteristics specific to these environments.

″General Performance″ includes performance tips and information on what occurs during product execution when running the IBM WebSphere Transformation Extender on all available platforms.

There are additional performance and execution characteristics specific to running the IBM WebSphere Transformation Extender under Batch and CICS on z/OS environments that can greatly affect their overall performance, including:

- Hardware Expectations and MIPS/MSU Ratings
- File Formats
- Pre-allocating Work Files (Temporary Data Sets)
- Paging in Memory
- External Factors

Hardware expectations and MIPS/MSU ratings

Hardware performance ratings such as MIPS or MSUs indicate machine-wide throughput, and not the speed of a single processor. As mentioned in ″Throughput and CPU Expectations″, a measurement of CPU throughput over multiple CPUs

might not accurately estimate the execution time of a single transformation. This results from the single-threaded nature of map executions.

Performance case

MIPS and MSU ratings account for all active processors on a particular hardware configuration, yet the Command Server or a single transformation can only use a single processor at any given time. Therefore, comparing MIPS or MSU ratings between machines to estimate transformation speed will not produce reliable estimates if the number of processors differs. To derive MSU ratings that can provide valid estimates, the number of processors must be taken into account.

Here is a hypothetical case using fictitious configurations to illustrate how estimating performance results might be affected by various factors. The considered factors include hardware configurations, MSU ratings, and taking into account the number of processors on the machine (normalizing) used to run maps or perform transformations.

Comparing performance

Analyze the following table of the MSU ratings for two different hardware configurations:

Configuration	Number of Processors	Rating (MSUs)
Config-02	two	28
Config-01	one	13

Conclusion of comparison

The MSU ratings and estimated MIPS ratings of the higher rated configuration, **Config-02**, imply that it yields twice the execution speed. This is not the case when measuring the execution time of a single map.

Performance case after normalizing

Normalizing a machine-wide MIPS rating for the number of processors on the machine results in a more representative estimate for comparative transformation performance. It accounts for the number of processors on a box.

Comparing performance

Analyze the following table of the MSU ratings that have been normalized for two different hardware configurations to account for the number of processors on each box:

Configuration	Number of Processors	Rating (MSUs) before Normalization	Rating (MSUs) after Normalization
Config-02	two	28	14
Config-01	one	13	13

Conclusion of comparison after normalization

Because the **Config-01** configuration has only a single processor, the normalized rating and non-normalized ratings are the same. This similarity in normalized

MSU ratings accounts for similarities in execution times. All other things being equal between the two configurations, the two different machines should execute a single transformation, *X*, in roughly the same time, *T*.

Parallel execution

Even though the transformation execution times for the two boxes are roughly the same, the **Config-02** configuration with two processors, could execute two instances of that same transformation *X* in parallel. This assumes that the input and output data sets have no interdependence. In this case, the execution time for each transformation is roughly the same.

If both transformations are initiated at the same time, however, they should complete in that same time interval, *T*. Because both transformations executed in parallel, the amount of transformation work for both is twice as much as it would be for a single execution.

Twice the amount of transformation work in the same amount of time yields double the throughput, which represents a significant performance improvement.

File formats

The z/OS operating system traditionally processes data in a record-oriented fashion. On other platforms (UNIX and Windows) it is byte (stream) oriented.

The z/OS operating system offers many file format choices, including record sizes, block sizes and record types for record-oriented processing.

Some of the most common record types are: Fixed or Fixed Block, and Variable or Variable Blocked record formats.

The IBM WebSphere Transformation Extender run under Batch and CICS environments provides an additional Record Separators execution command (`/V`) to enable special handling of data sets with *variable length* record formats, if required.

The special handling of record separators for variable length records occurs during initialization, finalization, or both depending upon your source, input data, and targets, output data.

Record separators (/V) execution command

For more efficient access during the transformation and validation processes, input files with a variable-length format are copied to temporary work files with the `/V` execution command-specified separators inserted directly into the file. This operation is reversed, and record separators removed, on targets where the output record format type is variable length.

Though from the Command Server processing perspective this is generally more efficient, there is a slight overhead cost of the copy-insertion-deletion operation to take into account for variable-length records

If your requirements can be met using a fixed format, then the use of Record Separator execution command (`/V`) will not be required along with its slight increased overhead.

If a data set of variable-length records already contains the separators defined to the map embedded in the data, this option should not be used.

See *z/OS Batch* and *z/OS CICS* for more information about the /V Record Separators execution command.

Pre-allocating work files (temporary data sets)

There are guidelines you can follow for pre-allocating work files before you run the IBM WebSphere Transformation Extender under Batch and CICS on z/OS environments.

Batch

The Batch Command Server, by default, and without explicit pre-allocation of work files, will attempt to allocate the work files dynamically during map execution.

Dynamic allocations incur a significant penalty compared to static pre-allocation of work files. Dynamic allocation inherits the overhead associated with the operating systems' dynamic allocation mechanisms. Therefore, pre-allocating work files before execution provides overall better performance.

A good guideline to use when pre-allocating work files is to allocate two temporary work files for each map card, plus one additional file. One of the two temporary work files for each card is used to process data sets containing variable-length records. The other temporary work file and the additional file are used as general-use work files.

See ″Temporary Data Sets (Work Files)″ for more details about allocating work files.

CICS

The IBM WebSphere Transformation Extender running under CICS uses a predefined VSAM workspace file for its work file processing requirements.

For more details about allocating work files, see ″Temporary Data Sets (Work Files)″.

Paging in memory

Processing work files in memory can improve some performance characteristics of a transformation execution. Introduced in the IBM WebSphere Transformation Extender for z/OS v6.7.2 release, two methods of processing the work file in memory are provided. They incorporate the **Work Files in Memory** and **Work Files in Memory - Hiperspace**™ execution commands.

WorkSpace Work Files in Memory

The **Work Files in Memory** execution command requests that the transformation engine handle all work file pages in address-space memory. An advantage in using this command is that it significantly reduces the I/O request volume involved in the transformation and therefore decreases EXCPs. This EXCP count reduction comes at a small CPU usage penalty ranging from 10% to 15% for some maps.

WorkSpace Work Files in Memory - Hiperspace

The **Work Files in Memory - Hiperspace** execution command requests that the transformation engine handle all work file pages in Hiperspace memory. An advantage in using this command is that it significantly reduces the I/O request

volume involved in the transformation and decreases EXCPs like the **Work Files in Memory** execution command, but at a lower cost of CPU usage.

Carefully consider using these execution commands to ensure that they do not monopolize your available operating system physical memory for both the **Work Files in Memory** and **Work Files in Memory - Hiperspace** execution commands. Indiscriminate use of these execution commands *might* result in excessive operating system paging. Use these execution command options on a case-by-case basis after consulting with your systems administration staff.

Valid **Work Files in Memory** execution commands for each execution environment are:

- Batch:
  - ″WorkSpace (Work Files in Memory) (-WM) or (/WM)″
  - ″WorkSpace (Work Files in Memory - Hiperspace) (-WH) or (/WH)″
- CICS
  - ″WorkSpace (Work Files in Memory) (-WM) or (/WM)″

External factors

External factors can greatly affect the overall performance of the IBM WebSphere Transformation Extender running under Batch or CICS on the z/OS operating environment and should be taken into account when assessing overall throughput and performance.

The following factors should be considered:

- Configuration of the operating system
- Work loads / Queue depths
- Work Load Manager policies and goals
- Batch-oriented scheduling issues relating to resource contentions or collisions, or both
- Transaction priorities

Some of these factors can significantly impact overall performance of the IBM WebSphere Transformation Extender running under Batch or CICS on the z/OS operating environment, thereby requiring a more in-depth analysis or real-time tools to gather resource statistics to isolate and take corrective action.

Tools such as RMF (Resource Measurement Facility), in conjunction with SMF (System Management Facilities) records, help in monitoring and identifying any performance problems within traditional batch-oriented processing. Additionally, the CICS Performance Monitor can be used to identify performance problems within CICS environments.

Batch execution environment

When executing the Batch Command Server, there are several areas where you might be able to improve performance results. These areas are:

- IBM Language Environment (LE) Runtimes
- Input Data Sets (Source)
- Output DataSets (Target)
- Temporary Data Sets (Work Files)

- Execution Command Settings
- Striped Data Sets

IBM Language Environment (LE) runtimes

The following recommendations are for the LE runtime library access modifications, which might improve the performance of batch jobs running under the Batch Command Server:

- **SCEERUN** and **SCEERUN2** runtime libraries into your **LNKLST**
- **SCEELPA** data set in your LPA list (LPALST*xx*)
- Heavily used modules in the Link Pack Area (LPA)

  If these IBM recommendations have not already been implemented in your system's environment, contact your systems' programming personnel to inquire about them.

  "Performance Considerations and CEEUOPT" discusses the runtime options module shipped with the IBM WebSphere Transformation Extender for z/OS and set specifically to avoid known performance-degrading options.

  See *Language Environment Customization* for additional information.

Performance Considerations and CEEUOPT

IBM WebSphere Transformation Extender for z/OS is shipped with a copy of the **CEEUOPT** runtime options module linked into the IBM WebSphere Transformation Extender executable module. This copy of **CEEUOPT** is included to ensure that known performance-degrading options cannot be included from the existing runtime environment.

The `STORAGE` initialization options cause the most notable performance degradation to IBM WebSphere Transformation Extender. `HEAPCHK` can also adversely affect the performance of IBM WebSphere Transformation Extender. `HEAP` initial size and increment and `STACK` initial size and increment might provide better IBM WebSphere Transformation Extender performance if tuned for a specific customer environment or set of maps.

A description of all of the **CEEUOPT** options can be found in the *Language Environment Customization Guide*, published by IBM. Also included in this guide are instructions about how to remove and replace the `CEEUOPT CSECT`. The `CEEUOPT CSECT` needs only be included in the **DTXCMDSV** module.

The copy of **CEEUOPT** used in IBM WebSphere Transformation Extender is coded as follows:

```
CEEUOPT  CSECT
CEEUOPT  AMODE ANY
CEEUOPT  RMODE ANY
         CEEXOPT HEAP=(1M,256K,ANY,FREE,8K,4K),                    X
               STACK=(128K,128K,ANY,KEEP),                         X
               ALL31=(ON),                                         X
               STORAGE=(NONE,NONE,NONE,8K),                        X
               HEAPPOOLS=(ON),                                     X
               TERMTHDACT=(UADUMP)
         END
```

The **CEEUOPT** is contained in the **DTXCMDSV** module in the runtime library.

To start tuning the **CEEUOPT** settings, you will need a IBM WebSphere Transformation Extender map, which does not perform well at your installation, and a representative sample of the data, which the map processes. Start by turning on the RPTSTG option RPTSTG=(ON), and then running Command Server with your map or maps, and data.

The RPTSTG option will produce a report showing the maximum amount of storage used for the **Language Environment** STACK and HEAP. Use these as initial values when you reassemble and link-edit your new CEEUOPT. The goal is to minimize the amount of STACK and HEAP allocations as these have a significant effect on the performance of IBM WebSphere Transformation Extender.

Note: Turn off the RPTSTG feature before your final performance-checks, as RPTSTG also impacts the performance of the **Language Environment** storage allocation modules.

Note: See the following IBM manual for specific details:
- *Language Environment Customization*

Input Data Sets (Source)

When running IBM WebSphere Transformation Extender maps using input (source) data sets, the following recommendation might improve the performance of batch jobs running under the Batch Command Server.

Verify that all input (source) data sets were originally created with an optimum block size. Optimum block sizes on IBM DASD devices are one half-track of data.

This verification should occur on a case-by-case (job-by-job) basis if you suspect there might be performance problems relating to running Batch Command Server jobs.

If through this verification process, you determine that the input (source) data sets do not have optimal block sizes, you can perform the following:

- Identify the creators of the data sets
- Confirm that re-blocking the input (source) data sets will not have any adverse affect on any other potential consumers of them
- Re-block the input (source) data sets

  Note: There might be instances where re-blocking the input (source) data sets might not be a viable option for various reasons including but not limited to external vendor specifications or other factors driving the application requirements.

  Note: See the following IBM manuals for specific details:
  - *DFSMS: Using Data Sets*
  - *DFSMS: Implementing System-Managed Storage*

Output data sets (Target)

When running IBM WebSphere Transformation Extender maps to create output (target) data sets, the following recommendation for output data set JCL data definition (DD) modifications might improve the performance of batch jobs running under the Batch Command Server.

- Enable the system to determine the block size on all physical sequential (DSORG=PS) output data sets (non-temporary) created using the Batch Command Server.
  - If your system is not running SMS, modify the REVERSEO (output data set) data definition (**DD**) that is in the DTXBMJCL JCL example included with the Batch Command Server installation as follows:

    ```
    //* Define the output data set
    //REVERSEO DD DSN=&MAPOUT,
    //            DCB=(DSORG=PS,RECFM=VB,LRECL=80,BLKSIZE=0),
    //            UNIT=&UNIT,
    //            SPACE=(TRK,(1,1),RLSE),
    //            DISP=(NEW,CATLG,DELETE)
    ```

  - The DSORG=PS, RECFM=VB and BLKSIZE=0 DCB specifications on the REVERSEO DD will ensure that the system determines the block size for optimum performance.

    See *DFSMS: Using Data Sets* and *DFSMS: Implementing System-Managed Storage*for additional information.

Temporary data sets (work files)

The Batch Command Server uses temporary data sets (work files), as described in *z/OS Batch* of the *Command Server* documentation. Temporary data sets are used throughout the JCL examples included in the Batch Command Server installation.

Temporary data set names must begin with &&, which indicates a temporary data set. They can use VIO or NON-VIO and can be statically or non-statically (dynamically) allocated.

When running IBM WebSphere Transformation Extender maps, the following recommendations for using temporary data sets might improve the performance of batch jobs running under the Batch Command Server:

- All required temporary data sets *should* be statically allocated using VIO, unless you are reusing work files.
- Static temporary data set allocations *must* be defined using RECFM=FBS (Fixed Block Standard).
- Static temporary data set allocations *should* specify LRECL=32760.
- Static temporary data set allocations should *not* specify a block size.

Using the recommendations stated above will provide the most optimal I/O performance for the Batch Command Servers' work files.

For an example JCL, see the **SYSTMP01** (temporary data set) data definition (**DD**) fragment from the DTXBMJCL JCL that is included in the Batch Command Server installation.

See the *Command Server* documentation for more information about statically allocated temporary data work files.

See *DFSMS: Using Data Sets* and the*DFSMS: Implementing System-Managed Storage* documentation for additional information.

Execution command settings

Execution command settings refer to those commands that are used in the Batch Command Server. They are documented in *z/OS Batch*.

Some performance improvements might be achieved with the execution commands used in the Batch Command Server.

WorkSpace Pagesize (Paging) (-Psize:count) or (/Psize:count)

Sets the page size and count. See the *Execution Commands* documentation for more details.

The execution region should always have enough space to hold all requested pages. The minimum region size (kilobytes) can be calculated by multiplying page size by page count (size * count). An even larger region size is preferable.

WorkSpace (Work Files in Memory) (-WM) or (/WM)

Sets the **WorkSpace** work files to reside in memory. See the *Execution Commands* documentation for more details.

WorkSpace (Work Files in Memory - Hiperspace) (-WH) or (/WH)

Sets the **WorkSpace** work files to reside in standard Hiperspaces. See *Execution Commands* documentation for more information.

Carefully consider the use of these execution commands to ensure that they do not monopolize your available operating system physical memory for both the **Work Files in Memory** and **Work Files in Memory - Hiperspace** execution commands. Indiscriminate usage of these execution commands *might* result in excessive operating system paging. Use these execution command options on a case-by-case basis after consulting with your systems administration staff. An **IEFUSI** installation exit can be installed in your system's environment. This exit can protect memory usage when using the **Work Files in Memory** execution command. However, it will *not* protect Hiperspace memory usage when using the **Work Files in Memory - Hiperspace** execution command.

Striped data sets

Striping allows you to divide data into blocks and place them in various partitions on several hard disks. Striping data sets makes it is possible to achieve performance improvements.

When running IBM WebSphere Transformation Extender maps, striping data sets might improve the performance of batch jobs running under the Batch Command Server.

- Examine your environment and determine if you can stripe sequential and VSAM data sets.
  - Data striping sequential and VSAM data sets can reduce the processing time required for batch jobs resulting in lowered EXCPS.

    Do not stripe temporary data sets (SYSTMP01-99) processed in IBM WebSphere Transformation Extender maps.

    For additional information, see *DFSMS: Using Data Sets* and *DFSMS: Implementing System-Managed Storage*.

XPLINK performance build

IBM WebSphere Transformation Extender provides two Batch Command Server load libraries: non XPLINK (default) and XPLINK (optional).

XPLINK (eXtra Performance Linkage) is an IBM performance optimization technique using new linkage conventions and is the recommended migration path for executing new and existing Batch Command Server jobs.

Performance improvements can be obtained when running the XPLINK load library version of the Batch Command Server.

For the latest information regarding the installation and usage of the Batch Command Server XPLINK performance build load library, see the z/OS Release Notes on the product Web site (www.ibm.com/software/integration/wtx).

CICS execution environment

When executing the IBM WebSphere Transformation Extender under CICS on the z/OS operating environment, there are several areas where you might be able to improve performance results. These areas are:

- IBM Language Environment (LE) Runtimes
- Temporary Data Sets (Work Files)
- Enqueuing on Data Files (-Q)
- Execution Command Settings
- Striped Data Sets
- CICS Shared Data Tables

  See "CICS Troubleshooting", *CICS System Definition Guide* and *Language Environment Customization*for additional information.

IBM Language Environment (LE) runtimes

IBM CICS requires LE runtime libraries to be installed and available to operate CICS. Language Environment can employ automatic storage tuning for CICS. Automatic storage tuning can improve the performance of LE applications running under CICS.

Automatic storage tuning is only available at certain levels of CICS Transaction Server.

See *CICS System Definition Guide* for additional information.

Temporary data sets (work files)

The IBM WebSphere Transformation Extender under CICS uses temporary data sets (work files) differently in CICS than in Batch.

When running IBM WebSphere Transformation Extender maps, the following recommendation for using temporary data sets, might improve the performance of CICS applications running the IBM WebSphere Transformation Extender under CICS:

- The CICS Execution Option uses a **VSAM RRDS** data set defined at product installation and configuration time for its workspace (temporary files) data set.
- Verify that the **CSDSTRNO** parameter correctly reflects the total number of **DATAFILS** configured *plus* the **MAP KSDS** in your system's environment. Incorrect settings can lead to string waits causing performance degradation.

  See *z/OS CICS* for detailed information about configuration requirements and options.

Enqueuing on data files (-Q)

The CICS Execution Option can use enqueuing on data files (-Q) as described in *z/OS CICS* of the *Command Server* documentation.

By default, I/O operations against inputs and outputs are *not* serialized. Therefore, it is recommended that you:

- Examine any potential sharing of inputs or outputs, or both, for maps that run concurrently, which could possibly necessitate usage of this option.

Execution command settings

Execution command settings refer to those commands that are used in the IBM WebSphere Transformation Extender under CICS. They are documented in *z/OS CICS*.

Some performance improvements might be achieved with the execution commands used in the IBM WebSphere Transformation Extender under CICS.

WorkSpace Pagesize (Paging) (-Psize:count) or (/Psize:count)

Sets the page size and count. See the *Execution Commands* documentation for more details.

WorkSpace (Work Files in Memory) (-WM) or (/WM)

Sets the **WorkSpace** work files to reside in memory. See the *Execution Commands* documentation for more details.

Carefully consider the use of these execution commands to ensure that they do not monopolize your available operating system physical memory for the **Work Files in Memory** execution command. Indiscriminate use of these execution commands might result in excessive operating system paging. Use these execution command options on a case-by-case basis after consulting with your systems administration staff.

Striped Data Sets

See ″Striped Data Sets″ for more information.

CICS shared data tables

CICS applications or transactions running the CICS Command Server and using **KSDS** outputs or **KSDS** inputs (source) can achieve some possible performance improvements by defining these data sets within **CICS Shared Data Tables**.

See the following IBM manuals for specific details:
- *CICS Shared Data Tables Guide*
- *CICS System Definition Guide*
- *CICS Resource Definition Guide*

Troubleshooting

This documentation lists some of the types of questions on performance optimization and helps identify some areas in the system's environment that can be modified to help achieve performance objectives. It is divided into the following environments:

- Batch Troubleshooting
- CICS Troubleshooting

Batch troubleshooting

The Batch Command Server can be subject to a wide variety of overall performance throughput. The performance factors considered here extend through:

- Usage and interpretation of the designed map or maps being executed by the Batch Command Server.
  - Internal functions used within the maps themselves.
- External factors that traditionally influence batch performance as a whole such as:
  - Input and output data sets and their corresponding block sizes.
  - Batch scheduling issues potentially impacting performance where it relates to potential resource contention and collisions.
  - Your system's environment's Workload Manager (WLM) policies and goals.

Some key areas targeted for improvement are:

- Excessive EXPS (I/Os)
- Excessive CPU Time Used

Excessive EXCPS (I/Os)

If your job executing a map or systems of maps seems to generate an excessive amount of EXCPS (Execute Channel Programs) (I/Os), there are a number of items you can immediately examine and correct to improve by reducing the total EXCPS.

The following table lists some recommendations to make better use of resources required for a particular map executed, along with the references to additional documentation:

Recommended Action
Documentation Reference For Batch Execution Environment

Identify and verify that the output data sets that your map (or maps) generates are being created using a system-determined block size.
Output Data Sets (Target)

Identify and verify that your input data sets are using optimum block sizes.
Input Data Sets (Source)

Statically allocate, use VIO for temporary data sets, set `LRECL=32760` and set `RECFM=FBS` on their definitions.
Temporary Data Sets (Work Files)

Calculate new `WorkSpace PageSize` *size:count* settings based upon your maps work. Increasing the amount of available paging memory can reduce EXCPs.
Execution Command Settings

Use striped data sets if applicable and available.
Striped Data Sets

Examine the usage of the -WM (Work Files in Memory) option.
Execution Command Settings

Examine the usage of the -WH (Work Files in Memory - Hiperspace) option.
Execution Command Settings

Proceed with any possible map tuning, if applicable.
Map Performance Tuning Tips

Install and run XPLINK load library version.
XPLINK Performance Build

Excessive CPU time used

If your job executing a map or systems of maps seems to be using an excessive amount of CPU (Central Processing Unit) time, there are a number of items you can immediately examine and correct to improve and reduce the total CPU time.

The following table lists some recommended actions you can take to make better use of valuable central processing resources required for a particular map execution, along with the reference to additional documentation:

Recommended Action
Documentation Reference For Batch Execution Environment

Calculate new `WorkSpace PageSize` *size:count* settings based upon your map(s) work. Execution Command Settings

Examine the usage of the -WH (Work Files in Memory - Hiperspace) option.
Execution Command Settings

Proceed with any possible map tuning, if applicable.
Map Performance Tuning Tips

Reduce EXCP count. Reducing the EXCP count can lower CPU time spent managing I/O.
Excessive EXCPS (I/Os)

External factors

In addition to the recommended actions in ″Batch Troubleshooting″, there are external factors that can influence the overall performance of the Batch Command Server.

The following table lists some of these factors along with suggested references.

External Factors
Reference

Batch scheduling collisions and resource contention

- SMF data and the associated RMF reports.
- Consult your systems' programming, operations or support personnel.

Workload Manager Policies and Goals

- Consult your systems' programming or support personnel.

CICS troubleshooting

The IBM WebSphere Transformation Extender running under CICS can be subject to a wide variety of overall performance throughput. The performance factors considered here extend through:

- Usage and interpretation of the designed maps being executed by the CICS Execution Option
  - Internal functions used within the maps themselves
- CICS application design and purpose as it relates to map usage and volume of data upon which is being operated
- External factors that traditionally influence CICS performance as a whole such as CICS:
  - Transaction mix
  - Journaling
  - Dispatching priority

Some key areas targeted for improvement are:

- Excessive CICS I/Os, Transaction CPU Time
- External Factors

Excessive CICS I/Os, transaction CPU time

If your CICS application or transaction, which is executing a map or systems of maps, seems to generate an excessive amount of EXCPS (I/Os) and your transaction itself is using an excessive amount of CPU time, there are a number of items you can immediately examine and correct if needed, to improve the total required EXCPS, and CPU time.

The following table lists recommendations to make better use of CPU resources required for a particular map execution, along with documentation that contains additional information:

Recommended Action
Documentation References

Identify and verify LSR (Local Shared Resources) pools, strings and usage of CICS shared data tables, if applicable.
IBM manuals:
- *CICS Performance Guide*
- *CICS Shared Data Tables Guide*
- *CICS System Definitions Guide*

Use striped data sets if applicable and available.
Striped Data Sets

Calculate new `WorkSpace PageSize` *size:count* settings based upon your map(s) work. Execution Command Settings

Examine the usage of the `-WM` (Work Files in Memory) option.
Execution Command Settings

Proceed with any possible map tuning, if applicable.
Map Performance Tuning Tips

External factors

In addition to the recommended actions in ″CICS Troubleshooting″, there are external factors that can influence the overall performance of the IBM WebSphere Transformation Extender running under CICS on the z/OS operating environment.

The following table lists some of these factors along with suggested references.

External Factors
Reference

Workload Manager Policies and Goals

- Consult your systems' programming or support personnel.

Other CICS performance information
IBM manual:

- *CICS Performance Guide*

Chapter 4. Glossary

Term	Definition	Term	Definition
Batch	Running in the background with no user assistance	**LRECL**	Logical Record Length
CICS	Customer Information Control System; online transaction processing program; running in the foreground communicating with users	**LSR**	Local Shared Resources
CPU	Central Processing Unit; contains the logic functionality and processes the instructions that drive a computer.	**Mainframe**	Large processors or servers
DCB	Data Control Block parameter of DD statement in JCL	**MIPS**	Million Instructions Per Second
DD	Data Definition statement in JCL	**MSU**	Measured Service Units
EXCP	Execute Channel Program	**MVS**	Multiple Virtual Storage; mainframe operating system
Hiperspace™	A virtual storage area (up to 2gigabytes in size) outside the user's virtual address space residing in real or expanded storage and backed up by auxiliary storage	**OS/390**	Mainframe operating system (was MVS) (old name - now referred to as z/OS)
I/O	Input and Output	**Systems Z**	Line of IBM computer systems running operating systems such as z/OS
LE	IBM Language Environment	**VIO**	Virtual Input/Output
LNKLST	Link List	**XPLINK**	eXtra Performance Linkage; An IBM performance optimization technique using new linkage conventions
LPA	Link Pack Area	**z/OS**	Mainframe operating system

Term	Definition	Term	Definition
RECFM	Record Format used in DCB parameter of DD statement in JCL		

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

AIX
AIX 5L
AS/400
Ascential
Ascential DataStage
Ascential Enterprise Integration Suite
Ascential QualityStage
Ascential RTI
Ascential Software
Ascential
CICS
DataStage
DB2
DB2 Universal Database
developerWorks
Footprint
Hiperspace
IBM
the IBM logo
ibm.com
IMS
Informix
Lotus
Lotus Notes
MQSeries
MVS
OS/390
OS/400
Passport Advantage
Redbooks
RISC System/6000
Roma
S/390
System z
Tivoli
Trading Partner

WebSphere
z/Architecture
z/OS
zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

IBM WebSphere Transformation Extender, Version 8.1

Index

S

T

W

X

Printed in USA