Informatica

Informatica® Intelligent Cloud Services
July 2024

Data Integration Performance Tuning

Informatica Intelligent Cloud Services Data Integration Performance Tuning
July 2024

Publication Date: 2024-10-15

Table of Contents

Preface

Use *Data Integration performance tuning* to learn how to optimize Data Integration mapping performance.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at http://www.informatica.com/cloud. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

Access the Informatica Intelligent Cloud Services Community at:

https://network.informatica.com/community/informatica-network/products/cloud-integration

Developers can learn more and share tips at the Cloud Developer community:

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and mapplets:

https://marketplace.informatica.com/

Data Integration connector documentation

You can access documentation for Data Integration Connectors at the Documentation Portal. To explore the Documentation Portal, visit https://docs.informatica.com.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at https://www.informatica.com/trust-center.html.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The Informatica Intelligent Cloud Services Status page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, on the Informatica Intelligent Cloud Services Status page, click **SUBSCRIBE TO UPDATES**. You can choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

You can contact a Global Support Center through the Informatica Network or by telephone.

To find online support resources on the Informatica Network, click **Contact Support** in the Informatica Intelligent Cloud Services Help menu to go to the **Cloud Support** page. The **Cloud Support** page includes system status information and community discussions. Log in to Informatica Network and click **Need Help** to find additional resources and to contact Informatica Global Customer Support through email.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at https://www.informatica.com/services-and-training/support-services/contact-us.html.

CHAPTER 1

Data Integration performance tuning overview

The goal of performance tuning is to optimize mapping performance by eliminating performance bottlenecks.

To tune the performance of a mapping, first identify a performance bottleneck, eliminate it, and then identify the next performance bottleneck until you are satisfied with the performance.

Determining the best way to improve performance is an iterative process. Change one variable at a time, and time the mapping both before and after the change. If the mapping performance doesn't improve, you might want to return to the original configuration.

Performance issues can occur for any of the following reasons:

- Best practices not followed
- Target bottlenecks
- Source bottlenecks
- Transformation bottlenecks
- System bottlenecks

Best practices

Follow best practices to optimize performance.

Consider employing best practices in the following areas:

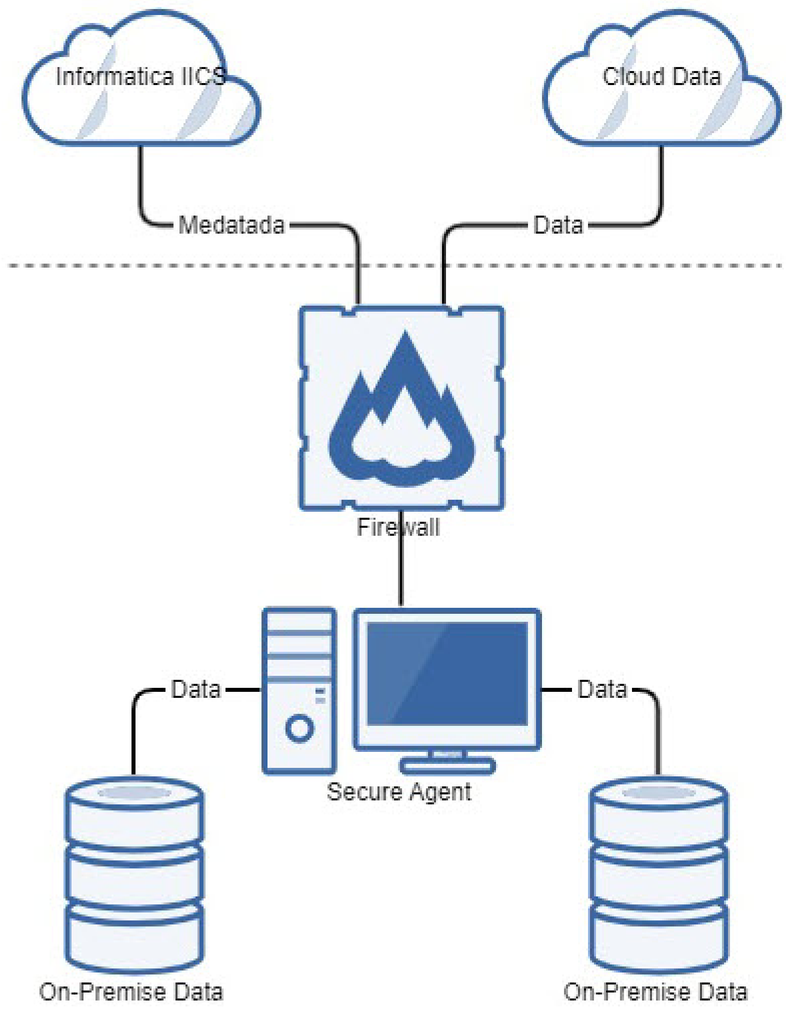
- Secure Agent environment and configuration
- SQL ELT optimization use
- Cloud connector performance
- Mapping design and environment

Secure Agent and cloud regions

To avoid performance impact due to network latency, locate the Secure Agent and the components it interacts with in the same region. For example, if your cloud data warehouse end points are located in an AWS US-West region, locate the Secure Agent machine within the US-West region.

If the Secure Agent and the components are deployed on the same cloud environment without on-premise components, put them on the same VPC/subnet. The availability zones should also be in the same region.

The following image shows the Informatica Intelligent Cloud Services Secure Agent and the components that it interacts with:



Secure Agent machines

For cloud data lake to cloud data warehouse mappings, choose hardware that supports high throughput in terms of disk input/output (I/O) and network bandwidth.

Be sure the machine that the Secure Agent is installed on meets the following requirements:

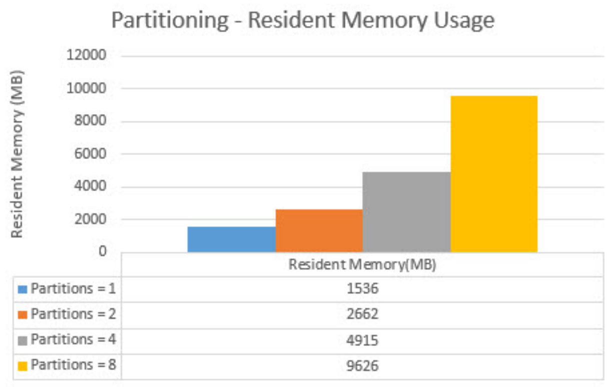
- Supports a minimum disk throughput of 500 MBps
- Supports a minimum network throughput of 2.5 Gbps

Secure Agent machine sizing requirements

Consider Secure Agent machine memory requirements for optimal performance.

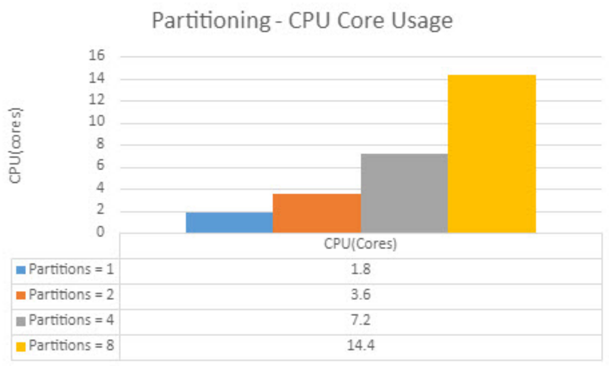
A typical cloud data lake to cloud data warehouse mapping might require up to 3 CPU cores and 1 GB of JVM heap memory for a data size of approximately 7.5 GB. The default value for JVM heap memory is 64 MB. Additional DTM buffer block sizing and buffer pool sizing will increase memory footprint.

The following graph illustrates the physical memory (resident memory) usage for a flat file to cloud data warehouse passthrough mapping as it relates to the number of partitions. The default buffer block size is set to 100 MB and the JVM heap memory is set to 1 GB.



	Resident Memory(MB)
🟦 Partitions = 1	1536
🟧 Partitions = 2	2662
⬜ Partitions = 4	4915
🟨 Partitions = 8	9626

Adding partitions to a cloud data lake to cloud data warehouse mapping linearly increases the requirement for CPU cores. The following graph illustrates the CPU consumption in cores for a flat file to cloud data warehouse passthrough mapping, with an increasing number of partitions.



	CPU(Cores)
🟦 Partitions = 1	1.8
🟧 Partitions = 2	3.6
⬜ Partitions = 4	7.2
🟨 Partitions = 8	14.4

To improve performance, configure the maxDTMProcesses custom property and the JVM options.

Secure Agent machine sizing requirements for advanced mode

For a Secure Agent that processes mappings in advanced mode, consider Secure Agent machine requirements for optimal performance.

At a minimum, the Secure Agent requires 4 CPUs, 16 GB of memory, and 100 GB of disk space. For optimal processing, use an SSD.

The following table lists the optimal Secure Agent configuration based on the number of concurrent Spark tasks that the Secure Agent processes:

Number of concurrent Spark tasks	CPU and memory	JVM heap size
0-250 Spark tasks	8 CPUs and 32 GB memory	2 GB
250-500 Spark tasks	16 CPUs and 64 GB memory On AWS, use 8 CPUs and 32 GB memory.	4 GB On AWS, use 2 GB.

The JVM heap size is set to 2 GB by default. Increase the heap size to avoid out of memory errors.

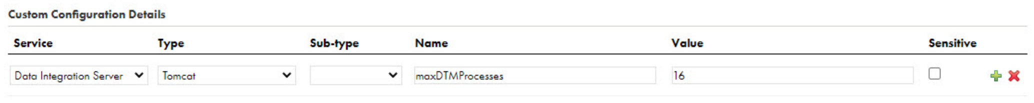
maxDTMProcesses property

Set the maxDTMProcesses custom property to improve performance.

By default, a Secure Agent can schedule two mapping tasks for execution. If there are more than two tasks, additional tasks are queued and then scheduled for execution when a slot becomes available. This can cause the Secure Agent machine's capacity to be underutilized.

To achieve better utilization of the CPU capacity of the Secure Agent machine and achieve a higher degree of concurrency, you can set the maxDTMProcesses custom property for the Data Integration Server to the number of parallel tasks. For example, setting this property to 16 allows 16 tasks to run simultaneously.

The following image shows configuration for the maxDTMProcesses custom property on the agent details page in Administrator:



Custom Configuration Details

Service	Type	Sub-type	Name	Value	Sensitive
Data Integration Server	Tomcat		maxDTMProcesses	16	☐

The recommended maxDTMProcesses value varies based on the connection types of the jobs that the agent runs:

- If the jobs use file-based or ODBC connections, use the following calculation: 0.75 times number of logical CPUs
  For example, if there are 24 logical CPUs on the Secure Agent machine, set the value to 18.
- If the jobs use cloud data lake or cloud data warehouse connectors, use the following calculation: 0.33 times number logical CPUs
  For example, if there are 24 logical CPUs on the Secure Agent machine, set the value to 8.

When you set the maxDTMProcesses custom property, note the following guidelines:

- Setting the property to a high value can help increase parallelism for task execution, but it can also cause performance bottlenecks in task execution time or increase job dispatch latency for mapping tasks.
- Setting the property to a lower value can lead to mapping jobs waiting for previous jobs to complete to acquire a slot.
- When you set the maxDTMProcesses property, don't exceed the terms of your license agreement. For example, if your organization is licensed for 3 Secure Agents and can run each agent on up to 4 CPUs, don't set this property to a value greater than 12.

Use these guidelines as a starting point and set the correct value iteratively.

For more information about setting agent properties, see *Runtime Environments*.

INFA_MEMORY and JVM options

You can configure the INFA_MEMORY property and JVM options to achieve optimal performance and avoid Java heap and memory-related errors. Define these properties on the agent details page in Administrator.

You can configure the INFA_MEMORY and JVM options using the following formats for values:

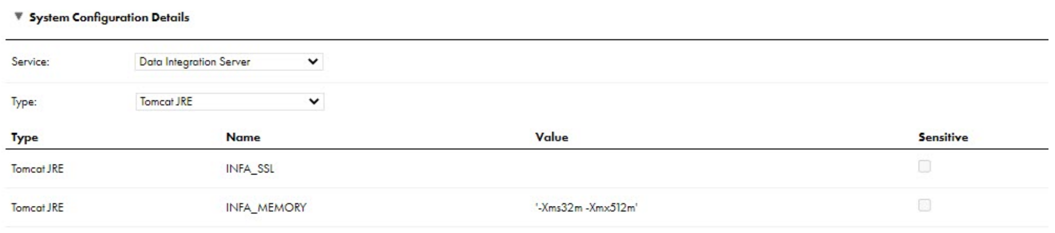
Format	Description
-Xms**m	The initial amount of memory allocated to the JVM when the Java process starts. Since this is the initial value, the Xms value can be small, for example, 64m or 128m. The Java process will allocate more space as required.
-Xmx****m	The maximum amount of memory that the JVM can allocate as heap. After the Java process starts, it will continue to allocate space to store its objects. The allocation can continue until the maximum setting is reached. Set the value to be large enough to hold all the Java objects and classes. On a 64-bit agent, the value can be about 1024M or 2048M.
-XX:MaxPermSize=***m	The maximum amount of permanent space that the JVM can use at a time. If the JVM needs more than the specified amount, the Java process fails. You can set this value to an average of 512M. However, the value must be less than the -Xmx value. Increase the MaxPermSize value if you receive an error about permGen space.

INFA_MEMORY

Define the INFA_MEMORY property on the agent details page in the **System Configuration Details** section. Select the Data Integration Service and the Tomcat JRE type, then define the INFA_MEMORY property.

The following image shows the INFA_MEMORY property on the agent details page:

Figure 1.



▼ System Configuration Details

Service: Data Integration Server

Type: Tomcat JRE

Type	Name	Value	Sensitive
Tomcat JRE	INFA_SSL		☐
Tomcat JRE	INFA_MEMORY	'-Xms32m -Xmx512m'	☐

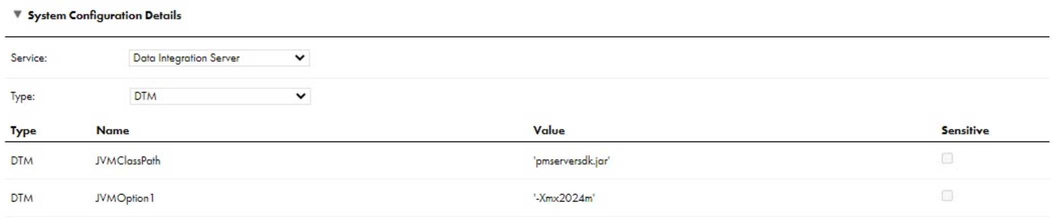
JVM options

Define JVM options on the agent details page in the **System Configuration Details** section. Select the Data Integration Service and the DTM type, then define a JVMOption property.

Each JVMOption property can include one JVM attribute. You can define up to five JVMOption properties. If you already have five JVMOption properties, you can create more as a custom property.

The -Xmx property determines the heap size. The default value is 64MB. For mappings that process large data volumes with multiple partitions, the default value can cause mapping failure because of insufficient Java heap space. The recommended value for this property is 2024MB.

The following image shows configuration for the JVMOption1 property on the agent details page:



Service: Data Integration Server

Type: DTM

Type	Name	Value	Sensitive
DTM	JVMClassPath	'pmserversdk.jar'	☐
DTM	JVMOption1	'-Xmx2024m'	☐

Secure Agent upgrades

If your organization uses multiple services and connectors, demand on a Secure Agent group can be high. To improve Secure Agent upgrade performance, you can disable services and connectors Secure Agent groups.

SQL ELT optimization

For a cloud data lake to cloud data warehouse integration, Data Integration intelligently identifies the opportunities to leverage cloud ecosystem or cloud data warehouse APIs for data pipeline processing.

Using SQL ELT optimization, Data Integration pushes the processing down to the cloud ecosystems or cloud data warehouses. This improves the data processing performance as data is processed close to the source and saves your company data transfer costs.

Data Integration supports full SQL ELT optimization for cloud data warehouse to cloud data warehouse integrations. Data Integration converts mapping logic to equivalent, optimized SQL queries that the cloud data warehouse executes, without data transfers that would normally incur additional processing time and data transfer charges. Data Integration uses the cloud data warehouse's compute capacity to process the data without additional resources, thus achieving compute efficiency.

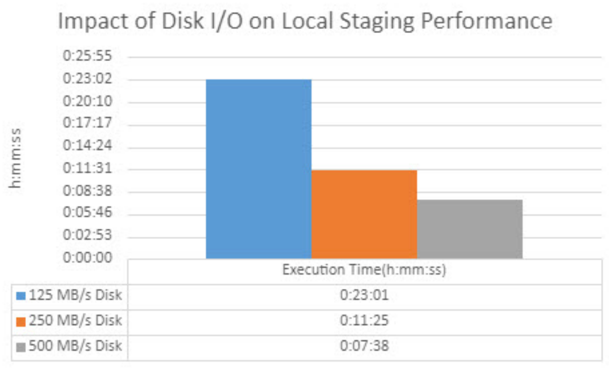
Try to use SQL ELT optimization for cloud data lake to cloud data warehouse or cloud data warehouse to cloud data warehouse integrations whenever SQL ELT optimization is supported.

Cloud connector performance

Cloud data lake and cloud data warehouse connectors are designed for optimal data loading and unloading performance.

A common design pattern across these connectors is in the way Informatica stages data locally on disk before uploading to an end point or after downloading data from an end point. This staging process is a disk-intensive operation and requires both CPU and disk I/O resources. Keep this in mind for all cloud data lake to cloud data lake, cloud data lake to cloud data warehouse, and cloud data warehouse to cloud data warehouse integrations when SQL ELT optimization isn't used.

The following graph represents the performance of a concurrent cloud data warehouse mapping and the impact that sustained disk I/O has on it:



	Execution Time(h:mm:ss)
🟦 125 MB/s Disk	0:23:01
🟧 250 MB/s Disk	0:11:25
⬜ 500 MB/s Disk	0:07:38

Informatica recommends a storage device with 500 Mbps disk throughput for a Data Integration workload with either partitioning enabled or concurrent executions. For detailed information on the various tuning options for these endpoints, see the appropriate connector guides. Performance tuning articles are available in the Informatica Knowledge Base for some connectors.

Mapping design and environment

When you design a mapping, follow best practices to optimize mapping performance.

Consider the following best practices:

Reduce data volume.

- Reduce the field precision of each column to the length that you need.
- Reduce the number of columns. Remove unconnected fields.
- Reduce the number of rows. Use a source filter in a Source or Filter transformation.
- Use a Filter transformation early in the mapping to remove unnecessary data.

Enable source partitioning.

Enable source partitioning whenever possible. The mapping task divides the source data into partitions and processes the partitions concurrently.

Optimize data conversion.

- Maintain consistency from source to target. Keep the same data type and precision across the pipeline whenever possible.
- For variable length data types, use precision as small as possible.
- Use the String data type instead of dates if no operations are done.
- Eliminate unnecessary data type conversions. For example, if a mapping moves data from an Integer column to a Decimal column, then back to an Integer column, the unnecessary data type conversion slows performance.
- Use integer values in place of other data types when performing comparisons using Lookup and Filter transformations wherever possible.

Use local flat file staging.

When a mapping writes to or reads from a cloud data warehouse, you can optimize the mapping performance by configuring the Secure Agent to stage data locally in a temporary folder before writing to or reading from the cloud data warehouse end point.

In the Secure Agent properties, set the staging property INFA_DTM_STAGING_ENABLED_CONNECTORS for Tomcat to the plugin ID of the cloud data warehouse connector. Data Integration creates a flat file locally to stage the data and then loads the data from the staging file to the data warehouse target or unloads data from the data warehouse source and stages it locally. For more information, see the individual cloud connector performance tuning guides.

Tune hardware.

For example, improve network speed and use multiple CPUs.

Consider the Secure Agent virtual machine instance type.

Choose performant cloud instances such as Amazon Elastic Compute Cloud (EC2), Azure Virtual Machine (Azure VM), or Google Cloud Platform (GCP) based on the resource requirements.

Bottlenecks

The first step in performance tuning is to identify performance bottlenecks. Performance bottlenecks can occur in the source and target databases, the mapping, the mapping task, and the system.

Tuning a bottleneck may reveal another one. The strategy is to identify a performance bottleneck, eliminate it, and then identify the next performance bottleneck until you're satisfied with the performance.

Look for performance bottlenecks in the following order:

1. Target
2. Source
3. Mapping
4. Mapping task
5. System

Use the following methods to identify performance bottlenecks:

Run test mappings.

You can configure a test mapping to read from a flat file source or to write to a flat file target to identify source and target bottlenecks.

Analyze thread statistics.

Analyze thread statistics to determine the optimal number of partitions.

Monitor system performance.

You can use system monitoring tools to view the percentage of CPU usage, I/O waits, and paging to identify system bottlenecks.

Thread statistics

You can use thread statistics in the session log to identify source, target, or transformation bottlenecks.

By default, a mapping uses one reader thread, one transformation thread, and one writer thread. The thread with the highest busy percentage identifies the bottleneck in the mapping run. When a specific transformation is identified as a bottleneck, you can tune the transformation.

The session log provides the following thread statistics:

Run time

Amount of time the thread runs.

Idle time

Amount of time the thread is idle, including the time the thread waits for other thread processing within the application. Idle time includes the time the thread is blocked by the Data Integration Server, but not the time the thread is blocked by the operating system.

Busy time

Percentage of the run time the thread is busy according to the following formula:

```
(Run time - idle time) / run time X 100
```

You can ignore high busy percentages when the total run time is short, such as under 60 seconds. This doesn't necessarily indicate a bottleneck.

To determine which transformation in the transformation thread is the bottleneck, view the busy percentage of each transformation in the thread work time breakdown. If the source, target, or transformations don't appear as the bottleneck, task properties could be causing the bottleneck.

CHAPTER 2

Optimizing targets

You can optimize performance by eliminating target bottlenecks. The most common performance bottleneck occurs when the task writes to a target.

Network latency, performance issues with the target instance type used for the database or data warehouse, or problems during heavy loading operations can cause target bottlenecks.

Identifying target bottlenecks

Test mapping performance using a flat file target and use thread statistics to identify a target bottleneck.

To identify a target bottleneck, complete the following tasks:

- Configure a copy of the mapping to write to a flat file target. If the performance increases significantly, you have a target bottleneck. If a mapping already writes to a flat file target, you probably don't have a target bottleneck.
- Read the thread statistics in the session log. When the mapping task spends more time on the writer thread than the transformation or reader threads, you have a target bottleneck.

Eliminating target bottlenecks

Best practices and target cloud database or data warehouse performance can affect the likelihood of target bottlenecks.

Complete the following tasks to eliminate target bottlenecks:

- Verify that you're following best practices for cloud and mapping design.
- Verify that the target cloud database or data warehouse is performing properly. If it's performing poorly, it can impact mapping performance.

CHAPTER 3

Optimizing sources

You can optimize performance by eliminating source bottlenecks. Performance bottlenecks can occur when the mapping task reads from a source.

Inefficient queries, network latency, or performance issues with the instance type used for the database or data warehouse source can cause source bottlenecks.

Identifying Source Bottlenecks

You can read the thread statistics in the session log to determine if the source is the bottleneck. When a mapping spends more time on the reader thread than the transformation or writer threads, you have a source bottleneck.

If the mapping reads from a cloud database or data warehouse source, use the following methods to identify source bottlenecks:

- Use a Filter transformation.
- Use a read test mapping.
- Use a database query.

If the mapping reads from a flat file source, you probably don't have a source bottleneck.

Using a Filter transformation

You can use a Filter transformation in the mapping to measure the time it takes to read source data.

Add a Filter transformation after each source. Set the filter condition to false so that no data is processed by the Filter transformation. If the time it takes to run the new mapping remains about the same, you have a source bottleneck.

Using a read test mapping

You can create a read test mapping to identify source bottlenecks. A read test mapping isolates the read query by removing the transformation in the mapping.

To create a read test mapping, complete the following steps:

1. Make a copy of the original mapping.
2. In the copy of the mapping, keep only the source transformations and any custom joins or queries.
3. Remove all other transformations.

4. Connect the sources to a flat file target.

Run the read test mapping. If the mapping performance is similar to the original run, you have a source bottleneck.

Using a query

To identify bottlenecks, run the read query directly against the source database or data warehouse.

Copy the read query directly from the session log. Run the query against the source with a query tool. On Windows, you can load the result of the query in a file. On Linux, you can load the result of the query in /dev/null.

Measure the query run time and the time it takes for the query to return the first row.

Eliminating source bottlenecks

Best practices and source cloud database or data warehouse performance can affect the likelihood of source bottlenecks.

Complete the following tasks to eliminate source bottlenecks:

- Verify that you're following best practices for cloud and mapping design.
- Verify that the source cloud database or data warehouse is performing properly. If it's performing poorly, it can impact mapping performance. The administrator can optimize the query to optimize performance.

CHAPTER 4

Optimizing mappings

You can optimize mapping performance by eliminating bottlenecks and following best practices when you design mappings.

Identifying mapping bottlenecks

If you determine that you don't have a source or target bottleneck, you might have a mapping bottleneck. Use thread statistics and Filter transformations to identify mapping bottlenecks.

To identify mapping bottlenecks, complete the following tasks:

- Read the thread statistics in the session log. When the mapping spends more time on the transformation thread than the writer or reader threads, you have a transformation bottleneck.
- Add a Filter transformation before each Target transformation. Set the filter condition to false so that no data is loaded into the target. If the time it takes to run the new mapping task is the same as the original mapping task, you have a mapping bottleneck.

Eliminating mapping bottlenecks

To eliminate mapping bottlenecks, optimize the mapping.

Mapping-level optimization may take time to implement, but it can significantly boost performance. Focus on mapping-level optimization after you optimize the targets and sources.

Generally, reduce the number of transformations in the mapping and delete unnecessary links between transformations to optimize the mapping. Configure the mapping with the least number of transformations and expressions to do the most amount of work possible. Delete unnecessary links between transformations to minimize the amount of data moved. Optimize the transformations that are in the mapping. Be sure to follow best practices for mapping design.

Optimizing delimited flat file sources

If a source is a delimited flat file, you specify the delimiter character to separate columns of data in the source file. You also specify the escape character.

Data Integration reads the delimiter character as a regular character if you include the escape character before the delimiter character. You can improve performance if the source flat file doesn't contain quotes or escape characters.

Optimizing data preview

Data preview performance depends on the number of rows that you preview. By default, the mapping previews 100 rows.

For sources that are larger than 1 GB, deselect both **Read Entire Source** and **Enable Upstream Preview**.

Optimizing filters

Use filters early in the data flow to improve mapping performance.

Use one of the following transformations to filter data:

- Source transformation. The Source transformation filters rows from database or data warehouse sources.
- Filter transformation. The Filter transformation filters data within a mapping. The Filter transformation filters rows from any type of source.

If you filter rows from the mapping, you can improve efficiency by filtering early in the data flow. Use a filter in the Source transformation to remove the rows at the source. The Source transformation limits the row set extracted from a database or data warehouse source.

If you can't use a filter in the Source transformation, use a Filter transformation and move it as close to the Source transformation as possible to remove unnecessary data early in the data flow. The Filter transformation limits the row set sent to a target.

Avoid using complex expressions in filter conditions. To optimize Filter transformations, use simple integer or true or false expressions in the filter condition.

Optimizing data type conversions

You can increase performance by eliminating unnecessary data type conversions.

For example, if a mapping moves data from an Integer column to a Decimal column, then back to an Integer column, the unnecessary data type conversion slows performance. Where possible, eliminate unnecessary data type conversions from mappings.

Use the following data type conversions to improve performance:

- Use integer values in place of other data types when performing comparisons using Lookup and Filter transformations. For example, many databases store U.S. ZIP code information as a Char or Varchar data type. If you convert the zip code data to an Integer data type, the lookup database stores the zip code 94303-1234 as 943031234. This helps increase the speed of the lookup comparisons based on zip code.
- Convert the source dates to strings through field-to-field conversions to increase performance. You can either leave the fields in targets as strings or change the fields to Date/Time fields.

Optimizing expressions

You can optimize the expressions used in transformations. When possible, isolate slow expressions and simplify them.

Complete the following tasks to isolate the slow expressions:

1. Remove the expressions one-by-one from the mapping.
2. Run the mapping to determine the time it takes to run the mapping without the transformation. If there is a significant difference in mapping run time, look for ways to optimize the slow expression.

Factoring out common logic

If the mapping performs the same task in multiple places, reduce the number of times the mapping performs the task by moving the task earlier in the mapping.

For example, you have a mapping with five target tables. Each target requires a Social Security Number lookup. Instead of performing the lookup five times, place the Lookup transformation in the mapping before the data flow splits. Next, pass the lookup results to all five targets.

Minimizing aggregate function calls

When writing expressions, factor out as many aggregate function calls as possible.

Each time you use an aggregate function call, the mapping searches and groups the data. For example, in the following expression, Data Integration reads COLUMN_A, finds the sum, then reads COLUMN_B, finds the sum, and finally finds the sum of the two sums:

```
SUM(COLUMN_A) + SUM(COLUMN_B)
```

If you factor out the aggregate function call as shown below, Data Integration adds COLUMN_A to COLUMN_B, then finds the sum of both:

```
SUM(COLUMN_A + COLUMN_B)
```

Replacing common expressions with local variables

If you use the same expression multiple times in one transformation, you can make that expression a local variable.

You can use a local variable only within the transformation. However, by calculating the variable only once, you speed up performance.

Choosing numeric instead of string operations

Mappings process numeric operations faster than string operations.

For example, if you look up large amounts of data on two fields, EMPLOYEE_NAME and EMPLOYEE_ID, configuring the lookup around EMPLOYEE_ID improves performance.

Choosing DECODE instead of LOOKUP

When you use a LOOKUP function, the mapping looks up a table in a database. When you use a DECODE function, you incorporate the lookup values into the expression so the mapping doesn't have to look up a separate table. When you want to look up a small set of unchanging values, use DECODE to improve performance.

Using operators instead of functions

Mappings read expressions written with operators faster than expressions with functions. Where possible, use operators to write expressions.

For example, you have the following expression that contains nested CONCAT functions:

```
CONCAT( CONCAT( CUSTOMERS.FIRST_NAME, ‘ ’) CUSTOMERS.LAST_NAME)
```

You can rewrite that expression with the || operator as follows:

```
CUSTOMERS.FIRST_NAME || ‘ ’ || CUSTOMERS.LAST_NAME
```

Optimizing IIF functions

IIF functions can return a value and an action, which allows for more compact expressions.

For example, you have a source with three Y/N flags: FLG_A, FLG_B, FLG_C. You want to return values based on the values of each flag. You use the following expression:

```
IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'Y', VAL_A + VAL_B + VAL_C,
  IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'N', VAL_A + VAL_B ,
    IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'Y', VAL_A + VAL_C,
      IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'N', VAL_A ,
        IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'Y', VAL_B + VAL_C,
          IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'N', VAL_B ,
            IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'Y', VAL_C,
              IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'N', 0.0,
)))))))
```

This expression requires 8 IIFs, 16 ANDs, and at least 24 comparisons.

If you take advantage of the IIF function, you can rewrite that expression as follows:

```
IIF(FLG_A='Y', VAL_A, 0.0)+ IIF(FLG_B='Y', VAL_B, 0.0)+ IIF(FLG_C='Y', VAL_C, 0.0)
```

This results in three IIFs, two comparisons, two additions, and a faster task run.

Optimizing window functions

Window functions process smaller frames faster. To reduce the size of the frame, decrease the number of rows between the frame offsets, and avoid including all preceding rows or all following rows.

Use a well-distributed partition key to create partitions of similar size.

Evaluating expressions

If you aren't sure which expressions slow performance, evaluate the expression performance to isolate the problem.

Complete the following steps to evaluate expression performance:

1. Time the mapping run with the original expressions.
2. Copy the mapping and replace half of the complex expressions with a constant.
3. Run and time the edited mapping.
4. Make another copy of the mapping and replace the other half of the complex expressions with a constant.
5. Run and time the edited mapping.

Optimizing Aggregator transformations

Aggregator transformations often slow performance because they group data before processing it. Aggregator transformations need additional memory to hold intermediate group results.

For optimal performance, place Aggregator transformations as close to the Source transformation as possible.

Use the following guidelines to optimize the performance of an Aggregator transformation:

- Group by simple columns.
- Use sorted input.
- Filter data before you aggregate it.
- Limit field connections.

Grouping by simple columns

You can optimize Aggregator transformations when you group by simple columns. When possible, use numbers instead of string and dates in the columns used for the GROUP BY. Avoid complex expressions in the Aggregator expressions.

Using sorted input

To increase mapping performance, sort data for the Aggregator transformation. Use the Sorted Input option to sort data.

The Sorted Input option decreases the use of aggregate caches. When you use the Sorted Input option, Data Integration assumes all data is sorted by group. As Data Integration reads rows for a group, it performs aggregate calculations. When necessary, it stores group information in memory.

The Sorted Input option reduces the amount of data cached during the task run and improves performance. To pass sorted data to the Aggregator transformation, use sorted input with either the Source transformation source filter or a Sorter transformation.

You can increase performance when you sort input in mappings with multiple partitions.

Filtering data before you aggregate

Filter the data before you aggregate it. If you use a Filter transformation in the mapping, place the transformation before the Aggregator transformation to reduce unnecessary aggregation.

Limiting connected fields

Limit the number of connected input/output or output fields to reduce the amount of data the Aggregator transformation stores in the data cache.

Optimizing Hierarchy Processor transformations

Hierarchy Processor transformations can slow performance based on the complexity of the hierarchical data.

Use the following guidelines to optimize the performance of a Hierarchy Processor transformation:

- When you read the hierarchical data in a Source transformation before passing it to the Hierarchy Processor transformation, use JSON format instead of an intelligent structure model whenever possible. However, use an intelligent structure model for files that are larger than 1 MB.
- Avoid non-splittable files that are larger than 100 MB. Instead, process several small JSON files.
- Reduce the number of nested hierarchies. Mappings process a three-level hierarchy faster than a five-level hierarchy.

Optimizing Joiner transformations

Joiner transformations can slow performance because they need additional space at run time to hold intermediary results.

Use the following guidelines to improve performance of a Joiner transformation:

Designate the master group as the source with fewer duplicate key values.

When Data Integration processes a sorted Joiner transformation, it caches rows for one hundred unique keys at a time. If the master group contains many rows with the same key value, Data Integration must cache more rows, and performance can be slowed.

Designate the master group as the source with fewer rows.

The Joiner transformation compares each row of the detail group against the master group. The fewer rows in the master, the fewer iterations of the join comparison occur, which speeds the join process.

Perform joins in a database or data warehouse when possible.

Performing a join in a database is faster than performing a join in the mapping. The type of database join you use can affect performance. Normal joins are faster than outer joins and result in fewer rows. In some cases, you cannot perform the join in the database, such as joining tables from two different databases or flat file systems.

Join sorted data when possible.

To improve mapping performance, configure the Joiner transformation to use sorted input. When you configure the Joiner transformation to use sorted data, Data Integration improves performance by minimizing disk input and output. You see the greatest performance improvement when you work with large data sets. For an unsorted Joiner transformation, designate the source with fewer rows as the master group.

Join the largest data set last.

In a mapping that has multiple Joiner transformations, join the largest data set in the most downstream transformation.

Set the broadcast join threshold.

Mappings in advanced mode perform a broadcast join for data sets that are smaller than the value set in the Spark session property `spark.sql.autoBroadcastJoinThreshold`. The mapping broadcasts the data set to all Spark executors across all advanced cluster nodes and reduces shuffle overhead for better performance.

Run CLAIRE Tuning to get a recommendation for the broadcast join threshold.

Optimizing Lookup transformations

If the lookup table is on the same database as the source table and caching isn't feasible, join the tables in the source database rather than using a Lookup transformation.

When you use a Lookup transformation, perform the following tasks to increase performance:

- Cache lookup tables.
- Optimize the lookup condition.
- Filter lookup rows.
- Index the lookup table.
- Optimize multiple lookups.

Caching lookup tables

If a mapping contains Lookup transformations, you might want to enable lookup caching.

When you enable caching, Data Integration caches the lookup table and queries the lookup cache during the task run. When this option isn't enabled, Data Integration queries the lookup table on a row-by-row basis.

The result of the lookup query and processing is the same, whether or not you cache the lookup table. However, using a lookup cache can increase mapping performance for smaller lookup tables. In general, you want to cache lookup tables that need less than 300 MB.

If you enable lookup caching, perform the following tasks to increase performance:

- Use the appropriate cache type.
- Enable concurrent caches.
- Optimize lookup condition matching.
- Reduce the number of cached rows.
- Override the ORDER BY statement.

- Use a machine with more memory.

Cache types

Consider the type of cache you use to improve performance.

You can use the following cache types:

Persistent cache

To save and reuse the cache files, you can configure the transformation to use a persistent cache. Use a persistent cache when you know the lookup table doesn't change between task runs. Using a persistent cache can improve performance because Data Integration builds the memory cache from the cache files instead of from the database.

Dynamic cache

Use a dynamic lookup cache to keep the lookup cache synchronized with the target. If the cache is static, the data in the lookup cache doesn't change as the mapping task runs. If the task uses the cache multiple times, the task uses the same data. If the cache is dynamic, the task updates the cache based on the actions in the task, so if the task uses the lookup multiple times, downstream transformations can use updated data.

Enabling concurrent caches

When Data Integration runs a mapping that contains Lookup transformations, it builds a cache in memory when it processes the first row of data in a cached Lookup transformation. If there are multiple Lookup transformations in a mapping, Data Integration creates the caches sequentially when the first row of data is processed by the Lookup transformation. This slows Lookup transformation processing.

You can enable concurrent caches to improve performance. When the number of additional concurrent pipelines is set to one or more, Data Integration builds caches concurrently rather than sequentially. Performance improves greatly when the tasks contain a number of active transformations that might take time to complete, such as Aggregator, Joiner, or Sorter transformations. When you enable multiple concurrent pipelines, Data Integration doesn't wait for active task runs to complete before it builds the cache. Other Lookup transformations in the pipeline also build caches concurrently.

Optimizing lookup condition matching

When the Lookup transformation matches lookup cache data with the lookup condition, it sorts and orders the data to determine the first matching value and the last matching value.

You can configure the transformation to return any value that matches the lookup condition. When you configure the Lookup transformation to return any matching value, the transformation returns the first value that matches the lookup condition. It doesn't index all fields as it does when you configure the transformation to return the first matching value or the last matching value. When you use any matching value, performance can improve because the transformation doesn't index on all fields.

Reducing cached rows

You can reduce the number of rows included in the cache to increase performance. Use the Lookup SQL Override option to add a WHERE clause to the default SQL statement.

Overriding the ORDER BY statement

By default, Data Integration generates an ORDER BY statement for a cached lookup. The ORDER BY statement contains all lookup fields.

To increase performance, suppress the default ORDER BY statement and enter an override ORDER BY with fewer columns.

Data Integration always generates an ORDER BY statement, even if you enter one in the override. Place two dashes (--) after the ORDER BY override to suppress the generated ORDER BY statement.

For example, a Lookup transformation uses the following lookup condition:

```
ITEM_ID = IN_ITEM_ID
PRICE <= IN_PRICE
```

The Lookup transformation includes three lookup fields used in the mapping, ITEM_ID, ITEM_NAME, and PRICE. When you enter the ORDER BY statement, enter the columns in the same order as the fields in the lookup condition. Enclose all database reserved words in quotes.

Enter the following lookup query in the lookup SQL override:

```
SELECT ITEMS_DIM.ITEM_NAME, ITEMS_DIM.PRICE, ITEMS_DIM.ITEM_ID FROM ITEMS_DIM ORDER BY
ITEMS_DIM.ITEM_ID, ITEMS_DIM.PRICE --
```

Using a machine with more memory

To increase performance, run the mapping on a Secure Agent machine with a large amount of memory. Increase the index and data cache sizes as high as you can without straining the machine.

If the Secure Agent machine has enough memory, increase the cache so it can hold all data in memory without paging to disk.

Optimizing the lookup condition

If you include more than one lookup condition, place the conditions in an optimal order to increase lookup performance.

Use the following order:

1. Equal to (=)
2. Less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=)
3. Not equal to (!=)

Filtering lookup rows

Create a filter condition to reduce the number of lookup rows retrieved from the source when the lookup cache is built.

Indexing the lookup table

Data Integration needs to query, sort, and compare values in the lookup condition columns. The index needs to include every column used in a lookup condition.

You can improve performance for the following types of lookups:

- Cached lookups. Index the columns in the lookup ORDER BY statement. The session log contains the ORDER BY statement.

- Uncached lookups. Index the columns in the lookup condition. Data Integration issues a SELECT statement for each row that passes into the Lookup transformation.

Optimizing multiple lookups

If a mapping contains multiple lookups, even with caching enabled and enough heap memory, the lookups can slow performance. Tune the Lookup transformations that query the largest amounts of data to improve overall performance.

Optimizing Machine Learning transformations

To optimize a Machine Learning transformation, configure bulk requests to combine multiple requests into one request that the transformation sends to the machine learning model.

Optimizing Normalizer transformations

A Normalizer transformation generates rows. To optimize performance, place the Normalizer transformation as close to the target as possible.

Optimizing Router transformations

When a Router transformation processes many output groups in a mapping in advanced mode, allow data to persist by setting the Spark session property `infaspark.sql.forcePersist=true` in the mapping task.

When data persists, the Router transformation doesn't repeat read operations for each output group.

Optimizing Sequence Generator transformations

To optimize Sequence Generator transformations, create a reusable sequence generator and use it in multiple mappings simultaneously. Also, configure the number of values to cache at one time.

Ensure that the number of values to cache isn't too small. Consider configuring the number of values to a value greater than 1,000.

If you don't need to cache values, set the number of values to cache to 0. Sequence Generator transformations that don't use a cache are faster than those that require caching.

When you connect the CURRVAL field in a Sequence Generator transformation, Data Integration processes one row in each block. You can optimize performance by connecting only the NEXTVAL field in a mapping.

Optimizing Sorter transformations

To optimize a Sorter transformation, allocate enough memory to sort the data and specify a different work directory for each partition in the transformation.

Allocating memory

For optimal performance, configure the Sorter cache size with a value less than or equal to the amount of available physical RAM on the Secure Agent machine.

Allocate at least 16 MB of physical memory to sort data using the Sorter transformation. The Sorter cache size is set to 16,777,216 bytes by default. If Data Integration can't allocate enough memory to sort data, the mapping task fails.

If the amount of incoming data is greater than the amount of Sorter cache size, Data Integration temporarily stores data in the Sorter transformation work directory. Data Integration requires disk space of at least twice the amount of incoming data when storing data in the work directory. If the amount of incoming data is significantly greater than the Sorter cache size, Data Integration may require much more than twice the amount of disk space available to the work directory.

Note: The session log contains the input row count and the size of incoming data for the Sorter transformation. For example, the following message appears when Data Integration processes the Sorter transformation:

```
SORT_40422 End of output from Sorter Transformation [srt_1_Billion_FF]. Processed
999999999 rows (866325637228 input bytes; 868929593344 temp I/O bytes)
```

In the message, the number of input rows is 999999999 and the size of the input rows is 866325637228.

Specifying a work directory

Data Integration creates temporary files when it sorts data. It stores them in a work directory. You can specify any directory on the Secure Agent machine to use as a work directory.

By default, Data Integration uses the value specified for the $PMTempDir service process variable. When you partition a mapping with a Sorter transformation, you can specify a different work directory for each partition in the pipeline. To increase mapping performance, specify work directories on physically separate disks on the Secure Agent machine.

To specify the work directory for a Sorter transformation, use the **Work Directory** advanced property in the Sorter transformation.

CHAPTER 5

Optimizing mapping tasks

You can optimize mapping task performance by identifying and eliminating bottlenecks.

Small cache size, low buffer memory, and small commit intervals can cause mapping task bottlenecks.

Identifying mapping task bottlenecks

To identify a mapping task bottleneck, analyze the performance details. Performance details display information about each transformation, such as the number of input rows, output rows, and error rows.

Eliminating mapping task bottlenecks

To eliminate mapping task bottlenecks, optimize the mapping task.

Buffer memory

When Data Integration initializes a task run, it allocates blocks of memory to hold source and target data.

Data Integration allocates at least two blocks for each source and target partition. Mapping tasks that use a large number of sources and targets might require additional memory blocks. If Data Integration can't allocate enough memory blocks to hold the data, the task fails.

You can configure the amount of buffer memory, or you can configure Data Integration to calculate buffer settings at run time.

To increase the number of available memory blocks, adjust the following mapping task properties:

- **DTM Buffer Size**. Increase the DTM buffer size advanced session property in the mapping task.
- **Buffer Block Size**. Decrease the buffer block size advanced session property in the mapping task.

Note: If data partitioning is enabled, the DTM buffer size is the total size of all memory buffer pools allocated to all partitions. For a task that contains *n* partitions, set the DTM Buffer Size to at least *n* times the value for the task with one partition.

Increasing DTM buffer size

The DTM buffer size setting specifies the amount of memory that Data Integration uses as DTM buffer memory. When you increase the DTM buffer memory, Data Integration creates more buffer blocks, which improves performance during momentary slowdowns.

Increasing DTM buffer memory allocation generally causes performance to improve initially and then level off. If you don't see a significant increase in performance, DTM buffer memory allocation isn't a factor in mapping performance.

To increase the DTM buffer size, open the task and edit the **DTM Buffer Size** advanced session property. Increase the DTM buffer size by multiples of the buffer block size.

Optimizing the buffer block size

If the Secure Agent machine has limited physical memory and the mapping contains a large number of sources, targets, or partitions, you might need to decrease the buffer block size.

If you're manipulating unusually large rows of data, increase the buffer block size to improve performance. If you don't know the approximate size of the rows, determine the row size by completing the following steps:

1. On the **Explore** page, open the mapping.
2. Open the Target transformation.
3. Click the **Target Fields** tab.
4. Add the precision for all columns in the target.
5. If you have more than one target in the mapping, repeat steps 2 - 4 for each additional target to calculate the precision for each target.
6. Repeat steps 2 - 5 for each source definition in the mapping.
7. Choose the largest precision of all the source and target precisions for the total precision in the buffer block size calculation.

The total precision represents the total bytes needed to move the largest row of data. For example, if the total precision equals 33,000, then Data Integration requires 33,000 bytes in the buffer block to move that row. If the buffer block size is only 64,000 bytes, then Data Integration can't move more than one row at a time.

To set the buffer block size, open the task and edit the **Default Buffer Block Size** advanced session property.

As with DTM buffer memory allocation, increasing buffer block size should improve performance. If you don't see an increase, then buffer block size isn't a factor in task performance.

Caches

Data Integration uses the index and data caches for XML targets and Aggregator, Rank, Lookup, and Joiner transformations.

Data Integration stores transformed data in the data cache before returning it to the pipeline. Data Integration stores group information in the index cache. Also, Data Integration uses a cache to store data for Sorter transformations.

To configure the amount of cache memory, specify the cache size. If the allocated cache isn't large enough to store the data, Data Integration stores the data in a temporary disk file, a cache file, as it processes the task data. Performance slows each time Data Integration pages to a temporary file.

Perform the following tasks to optimize caches:

- Limit the number of connected input/output and output only fields.

- Increase the cache sizes.

Limiting connected fields

For transformations that use data cache, limit the number of connected input/output and output only fields. Limiting the number of connected input/output or output fields reduces the amount of data the transformations store in the data cache.

Increasing cache sizes

Configure the cache size to specify the amount of memory allocated to process a transformation. The amount of memory you configure depends on how much memory cache and disk cache you want to use.

If the cache size isn't big enough, Data Integration processes some of the transformation in memory and pages information to cache files to process the rest of the transformation. Each time Data Integration pages to a cache file, performance slows.

If the mapping contains a transformation that uses a cache, and you run the task on a machine with sufficient memory, increase the cache sizes to process the transformation in memory.

Verbose logs

You can run a mapping task in standard or verbose execution mode. When you run the task in verbose execution mode, the mapping generates additional data in the logs that you can use for troubleshooting.

Use verbose execution mode only for troubleshooting purposes. Verbose execution mode impacts performance because of the amount of data it generates.

Best practices in advanced mode

When you create a mapping task based on a mapping in advanced mode, run CLAIRE Tuning to get recommendations to improve job performance. CLAIRE recommends settings for the Spark driver, Spark executors, and other Spark session properties.

CHAPTER 6

Optimizing advanced clusters

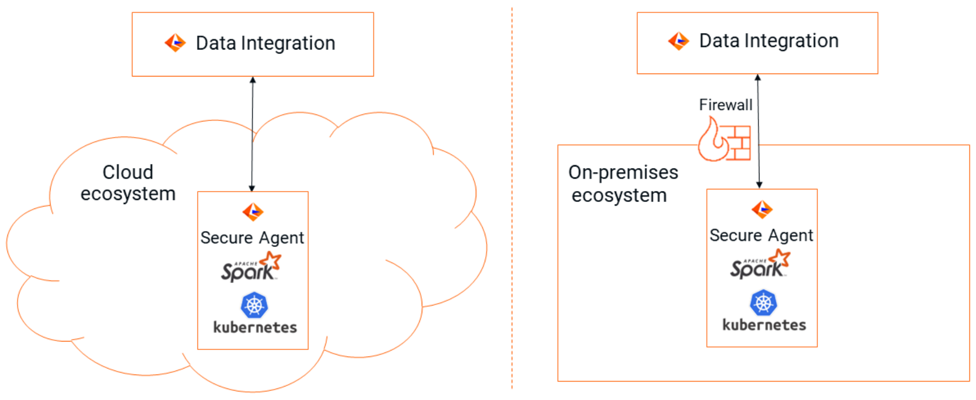
You can optimize advanced cluster performance by provisioning enough resources to process your workload.

Advanced cluster components

Data Integration interacts with advanced cluster components based on the cluster type.

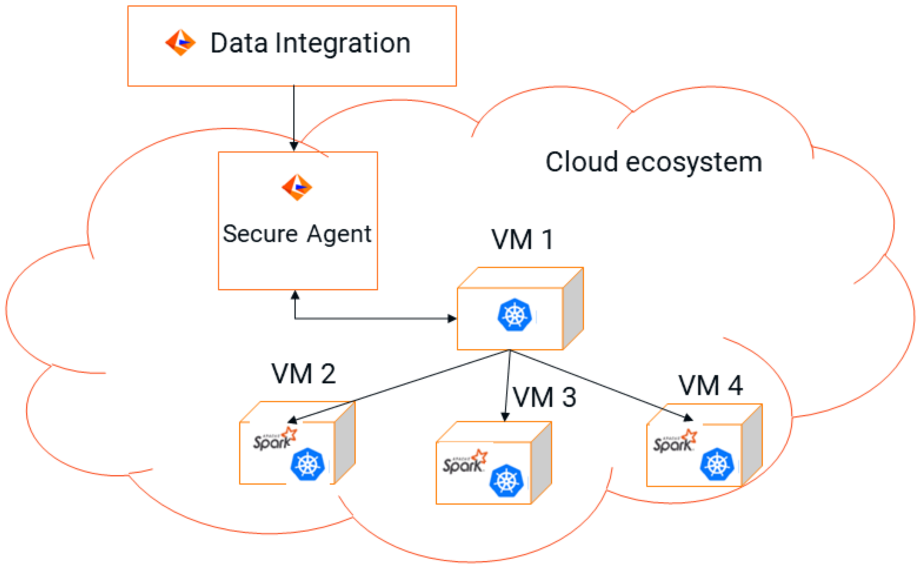
Local cluster

The following image shows how Data Integration interacts with the Secure Agent and advanced cluster components in a local cluster for both cloud and on-premises ecosystems:



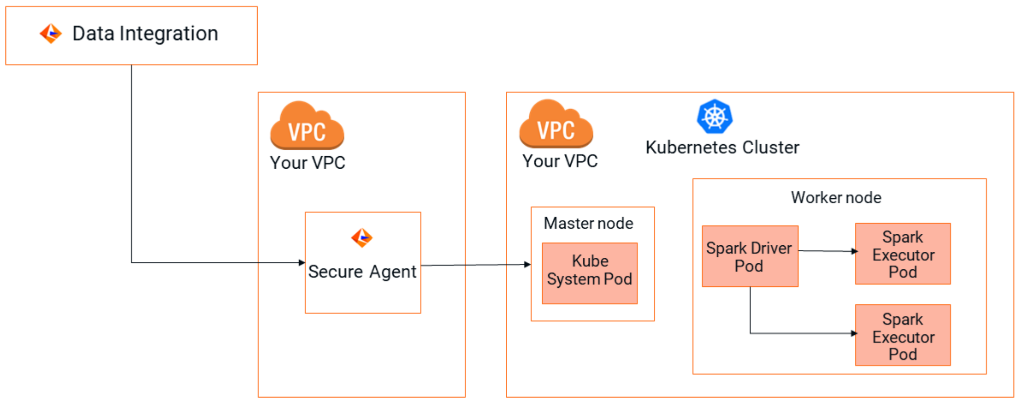
Fully-managed cluster

The following image shows how Data Integration interacts with the Secure Agent and advanced cluster components in a fully-managed cluster:



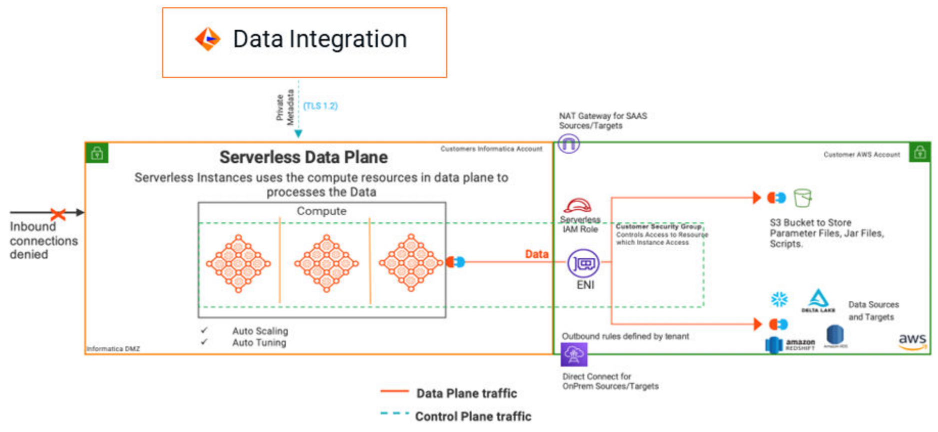
Self-service cluster

The following image shows how Data Integration interacts with the Secure Agent and advanced cluster components in a self-service cluster:



Advanced cluster in a serverless runtime environment

The following image shows how Data Integration interacts with advanced cluster components in a serverless runtime environment:


Data Integration
Private Metadata
(TLS 1.2)
NAT Gateway for SAAS Sources/Targets
Serverless Data Plane
Customers Informatica Account
Serverless Instances uses the compute resources in data plane to processes the Data
Compute
Inbound connections denied
Data
Auto Scaling
Auto Tuning
Informatica DMZ
Customer AWS Account
Serverless IAM Role
Customer Security Group
Controls Access to Resource which Instance Access
ENI
S3 Bucket to Store Parameter Files, Jar Files, Scripts.
Data Sources and Targets
Outbound rules defined by tenant
Direct Connect for OnPrem Sources/Targets
Data Plane traffic
Control Plane traffic

Best practices

When you create an advanced configuration, follow best practices to optimize advanced cluster performance.

Consider the following best practices:

Enable storage auto-scaling.

Use storage auto-scaling to dynamically change the amount of disk space that is available to process jobs. Jobs require disk space based on the data logic and the data volume in the job.

Enable worker node auto-scaling.

Use worker node auto-scaling to dynamically change the number of nodes that are available to process jobs.

For more information about auto-scaling, see the following Informatica blog: Cloud Data Integration Elastic - Understanding Auto Scaling.

Use Spot Instances.

Spot Instances provide the same performance as On-Demand Instances but at a lower price. However, they might not always be available. Use Spot Instances with a frequency of interruption that is less than 5%. For a list of Spot Instances and their frequency of interruption, see AWS's Spot Instance advisor.

Development and QA environments can use Spot Instances to save costs during internal tests and debugging. Avoid Spot Instances for jobs that have a strict SLA.

Optimizing advanced cluster nodes

You can optimize master and worker nodes by provisioning enough resources for each node type.

Optimizing master nodes

Provision a master node with enough resources to manage the worker nodes in the advanced cluster.

The following table lists the master node configuration based on the number of worker nodes:

Worker node count	Master node configuration
1-10	4 CPUs and 8 GB memory
11-100	8 CPUs and 32 GB memory

Optimizing worker nodes

Provision worker nodes with enough resources to process jobs.

The following table lists the worker node configuration based on the advanced cluster type:

Advanced cluster type	Worker node configuration
Fully-managed cluster	Minimum of 8 CPUs and 32 GB memory. For better performance, use 16 CPUs and 64 GB memory.
Local cluster	Minimum of 4 CPUs and 16 GB memory. For better performance, use 8 CPUs and 32 GB memory.

The number of worker nodes that you provision depends on your SLA.

Optimizing instance types

Select worker node instance types based on the data logic that you process on the advanced cluster.

Using GPU instances

GPU instances can provide a 5x performance gain and 72% lower TCO. However, a significant number of operations must be able to run on GPU.

To find out which operations run on GPU, use the Spark event log and search for GPU-CPU data exchange operations such as GPUColumnarToRow and GPURowToColumnar.

Using Graviton instances

Graviton2 instances in an advanced cluster on AWS can be up to 26% faster for CPU-intensive jobs and 41% cheaper. Shuffle-intensive jobs that include the Aggregator, Joiner, Rank, and Sorter transformations might not see a difference.

Using AMD chipsets

AMD chipsets, such as AMD EPYC 7452, on master and worker nodes in an advanced cluster on Microsoft Azure can be 1.2x faster than Intel Xeon. Shuffle-intensive jobs that include the Aggregator, Joiner, Rank, and Sorter transformations as well as mappings with complex expressions can be 1.3-1.4x faster.

CHAPTER 7

Optimizing system performance

After you tune the source, target, mapping, and mapping task, consider tuning the system to prevent system bottlenecks and optimize performance.

Data Integration uses system resources to process transformations, run tasks, and read and write data. Data Integration also uses system memory to create cache files for transformations such as the Aggregator, Joiner, Lookup, Sorter, and Rank transformations.

Identifying system bottlenecks

You can use system tools to monitor Windows and Linux systems.

Identifying system bottlenecks on Windows

You can view the **Performance and Processes** tab in the Task Manager for system information. The **Performance** tab in the Task Manager provides an overview of CPU usage and total memory used.

Use the Performance Monitor to view more detailed information. The following table describes the system information that you can use in the Windows Performance Monitor to create a chart:

Property	Description
Percent processor time	If you have more than one CPU, monitor each CPU for percent processor time.
Pages/second	If pages/second is greater than five, you may have excessive memory pressure known as thrashing.
Physical disks percent time	The percent of time that the physical disk is busy performing read or write requests.
Physical disks queue length	The number of users waiting for access to the same disk device.
Server total bytes per second	The server has sent to and received from the network.

Identifying system bottlenecks on Linux

Linux provides several tools that you can use to identify system bottlenecks.

You can use the following commands:

- top. View overall system performance. This tool displays CPU usage, memory usage, and swap usage for the system and for individual processes running on the system.
- iostat. Monitor the loading operation for every disk attached to the database server. Iostat displays the percentage of time that the disk is physically active. If you use disk arrays, use utilities provided with the disk arrays instead of iostat.
- vmstat. Monitor disk swapping actions.
- sar. View detailed system activity reports of CPU, memory, and disk usage. You can use this tool to monitor CPU loading. It provides percent usage on user, system, idle time, and waiting time. You can also use this tool to monitor disk swapping actions.

Eliminating system bottlenecks

After you identify system bottlenecks, tune the system to remove the bottlenecks and improve performance.

Complete the following tasks to eliminate system bottlenecks:

- If the CPU usage is more than 80%, check the number of concurrent running tasks. Consider changing the load or using a Secure Agent group to distribute tasks to different agent machines. If you can't reduce the load, consider adding more processors.
- If swapping occurs, increase the physical memory, or reduce the number of memory-intensive applications on the disk.
- If you have excessive memory pressure (thrashing), consider adding more physical memory.
- If the percent of time is high, tune the cache for transformations to use in-memory cache instead of writing to disk. If you tune the cache, requests are still in queue, and the disk busy percentage is at least 50%, upgrade to a faster disk. If physical disk queue length is greater than two, consider upgrading the disk.
- If the percent time spent waiting on I/O is high, consider upgrading to a faster disk or using other under-utilized disks. For example, if the source data, target data, lookup, rank, and aggregate cache files are all on the same disk, consider putting them on different disks.

INDEX