International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

Int. J. Sci. Res. Arch.
IJSRA
International Journal of Science and Research Archive
Research Journal Archive, INDIA

(Review Article)

Check for updates

Comprehensive tuning guide: IBM MDM, PME algorithm, services, and configurations

Sai Reddy Anugu *

Individual Researcher.

International Journal of Science and Research Archive, 2024, 13(02), 043–047

Publication history: Received on 23 September 2024; revised on 27 October 2024; accepted on 31 October 2024

Article DOI: https://doi.org/10.30574/ijsra.2024.13.2.2097

Abstract

The research paper provides comprehensive guidance on tuning specifics to maximize the efficiency of IBM MDM, particularly in high-performance environments. It addresses several advanced topics and tuning guidelines, focusing on the architectural framework and the details required for sustained performance. Including the recommendations to Handle Long-Chained Duplicates in Suspect Duplicate Processing (SDP), Minimize Deadlocks During Duplicate Processing, and Advanced Database Configurations for High-Performance WebSphere Application Server (WAS) Java Virtual Machine (JVM) and Cache Optimization, and Thread and session management, upgrading the Java version to hava8, Garbage Collection (GC) and Heap Memory Settings, servlet caching, and reduced application logging, MDM Workbench recommendations, Services tuning, and Batch processor tuning. IBM Messaging Queue's Max server connections and Queue Connection Factory Settings.

Keywords: Master Data Management (MDM); Probabilistic Matching Engine (PME); Duplicates in Suspect Duplicate Processing (SDP); WebSphere Application Server (WAS); Event Manager (EM)

1. Introduction

This paper provides the tuning recommendations to help organizations maximize MDM's capabilities to meet current and future data management requirements. It outlines the critical focus areas, including tuning for MDM Workbench, services, WebSphere Application Server (WAS), database configurations, and messaging queue configurations and helping organizations maximize MDM's capabilities to meet current and future data management requirements. Emphasizes the goal of creating a scalable and high-performance MDM environment equipped to handle complex data management tasks and improve transaction throughput while minimizing system conflicts and resource utilization.

2. MDM Workbench Tuning

2.1. Enable Override Base Query

This feature ensures a fine-grained transaction uses single SQL select/insert/update statements when interacting with an extension to an out-of-the-box entity. From Knowledge Center: Override base query: Select this option to optimize the SQL queries for the data extension. This option requires manual coding and should not be used for entities that support pluggable SQL. See Generated code for data extensions for additional information.

Note: it is necessary to manually edit the Java code generated by the "Override base query" feature to modify the generated code so that one SQL SELECT is used instead of two. However, this feature ensures that the add/update SQL is done through one call (so it is helpful to enable this feature, even if you do not subsequently modify it to handle the get call). Pluggable SQL is used by these entities: Person, Organization, Contract, and ProductInstance. When you create data extensions, additional associated code is generated to support the created data extensions. Pluggable SQL is

* Corresponding author: Sai Reddy Anugu

enabled if you create a data extension for a person, organization, contract, product instance, or any of their child objects. You should use the updateInqLevel administration transaction to update the SQL for affected queries. The query SQL in the generated code for the data extension will be ignored for transactions that use pluggable SQL. However, pluggable SQL is only used by coarse-grained transactions, like getContract. Ensure that the updateInqLevel transaction updates the inquiry level of the group that corresponds to the group to which your extension applies. If you have extended ContractComponent, for example, when you use the getContractComponent transaction, the Workbench will create the SQL used for this entity. In that case,

enabling the Override base query for all child entities (assuming you use fine-grained transactions to access them) is best. Enabling "Override Base Query" will improve add/update/get transaction performance. However, the performance gain will not be huge, especially for the get. The development effort should be considered when the entity is being modified – perhaps when adding another attribute – at this point, enabling the "Override Base Query" feature can be considered.

2.2. Use Inline Extension for Data Extension

In IBM MDM, we can implement the data extensions in two ways: side table extension and inline Extension. The recommended approach is to use the Inline Extension for better performance, lower CPU utilization, fewer inserts and updates while persisting data and fewer tables to join when querying the data.

3. Services Tuning

Typically, in a project, when the ETL or middleware tools send the request to MDM, it is recommended that the response be reduced to the absolute minimum.

- Enable smart Inquires
- Use the optimized inquiry levels in inquiry services
- Implement the pagination feature in search and get services
- Use optimized inquiry levels while retrieving the data
- Maintain the combination of coarse and fine-grain services for minimal network calls.

To satisfy extremely demanding non-functional requirements to retrieve relatively simple/small data sets, it is widespread for MDM projects to allow consumers to access the MDM database directly (though JDBC calls into the database, for example). The overhead of making calls through the J2EE layer can be relatively high (compared to the time taken for a direct database call, for example). So, a pragmatic and balanced approach can be taken between using the J2EE layer for add/updates/complex gets + searches while using direct SQL and prepared statement calls to the database for simple, fast gets and lookups.

4. Tuning Batch processor

- Initial load was performed by turning off the derived data synchronization as there was a defect in the OOTB MDM (i.e., to avoid the Deadlocks on DB2).
- The batch logs are re-directed to the MDM landing zone to avoid the space issue in log4j.properties.
- No of the submitters are adjusted to 100 to utilize more CPU.
- Submitter.number = 100
- deadlockMaxRetries = 5
- To avoid incoming deadlocks or successive duplicate records, this value may be used to randomize the order in records from the input passed on for processing. Each block of 'x' records is read from input as a group and then passed on randomly. Only blocks of records are randomized on the assumption that all 'x' records will be completed before processing the next record. Set to 0 or less to turn off randomization
- RandomizedWindowSize = 300
- Enabled the following properties to Dynamically change the submitter number for Batch based on the CPU availability without Restarting the Batch
- maximumThroughput = 0
- throughputSampleTime = 5
- throughputWindowSize = 2
- throughputReportingPeriod = 2
- Note: This option did not work well, so we disabled it.

5. PME Tuning

To improve the efficiency of the Suspect Duplicate processing, the following Items need to be considered and handled.

- How Many active persons before de duplication?
- How many organizations before de-duplication?
- How many 1:1, 1:2,1:3, and 1:n chains of duplicates?

Select count(cont_id) as cnt, cont_id from suspect where inactivated_dt is null and CUR_SUSPECT_TP_CD= 1 group by cont_id order by cnt desc fetch first 50 rows only

Few queries to verify the bucket sizes

SELECT count (BKTHASH) as cnt, BKTHASH FROM EME_RECBKTD GROUP BY BKTHASH order by cnt desc fetch first 20 rows only

Customize the algorithm to add or remove to match the customer's requirements.

Load one-third of the production data profile and analyze the data by conducting the match pair Analysis Activity with a group of people, Analyzing the results and findings, testing the new scores, and finalizing the recommendations.

Large buckets will cause performance issues. To avoid performance bottlenecks, set the maximum bucket size, which tells the PME not to create a bucket if this value is present in a record. We must run a few bucketing analysis jobs in an MDM Virtual deployment (part of the MDM Workbench + MDM Windows install) to collect and include the large bucket stats into the algorithm. And add anonymous values to filter the common values from bucketing.

6. Use Batch processor over Event Manager (EM)

Use of the Batch processor over Event Manager (EM) is recommended for the following reasons:

- Easy to stop/re-start Batch with different submitters, properties, and retries in case of deadlocks.
- I have found it hard to stretch MDM using Event Manager entirely (in case of a large chain of duplicates, Event Manager uses the collapsePartiesWithRules transaction to collapse, which is not an efficient way of collapse)
- It is tough to know precisely how far EM is in progress.
- Batch gives you immediate and obvious feedback on TPS and errors (for every 1000 transactions).

7. Handling the long-chained duplicates

A1 suspects are using collapseMultipleParties via Batch or ETL and are not using collapsePartiesWithRules via Event Manager. collapsePartiesWithRules will struggle since it recursively processes chains of matched parties when trying to process those long chains of match parties: this is not the most efficient way of dealing with large chains of A1 parties during auto-collapse.

8. Minimize the number of suspects to be processed by the data stewards

To minimize the number of deadlocks during the collapse process of Suspect Duplicate Processing, it is recommended the following:

- Perform Database reorgs at regular intervals.
- Ensure the indexes are moved to larger page sizes (32k)
- Change lock size row to Lock size Any (Page)
- Switch off Track mode for Image copies.
- Change the ZPARM SKIPUNCI=NO to SKIPUNCI=YES to avoid deadlocks in DB.
- To pin all 4 EME_RECBKTD, EME_RECCHKD, EME_RECCCMPD, EME_RECHEAD tables to memory.
- Ensure Index hit ratio> 95%
- Ensure Data hit ratio >80%

- Ensure Bufferpool memory equals 20-30% of database size (Perform Bufferpool "hit" Analysis: typically increase Bufferpool until index hit ratios max out.)
- Pin the SUSPECT Table to the memory. And remove the database triggers for the H_SUSPECT Table (unless required for specific use-case).

9. Application Server Tuning (WAS)

- Increase prepared statement cache size
- JDBC providers > DB2 Universal JDBC > Data sources > * > data source properties
  - Statement cache size = 300
- Increase EJB cache size
- Server clusters > mdm-cluster > Cluster members > mdm-server > EJB cache settings
  - Cache size = 3500
- Increase JDBC connection pool size to support parallelism
- JDBC providers > DB2 Universal JDBC > Data sources > * > Connection pools
  - Minimum connections = 50
  - Maximum connections = 200
- Use initial/maximum heap settings as 1024M/2048M to start
- Server clusters > mdm-cluster > Cluster members > mdm-server > Class loader > Process Execution >Server Component > Process Definition > Java Virtual Machine
  - Initial Heap Size = 1024
  - Maximum Heap Size = 2048
- Default GC policy is good enough if SDP is disabled. With SDP enabled, gencon GC gives higher throughput. However, in that case, keep switching off WebSphere PDA to avoid periodic JVM heap dumps.
- Deselect "prefer local" (for optimum Batch performance to distribute the requests to all the nodes and JVMs)
- Reduce the logging: Ensure the logging level for production is set to ERROR • Do not have System.out.println() statements in custom code.
- AIX Parameter changes on WAS
  - AIXTHREAD_COND_DEBUG = OFF
  - AIXTHREAD_COND_RWLOCK = OFF
  - AIXTHREAD_MUTEX_DEBUG = OFF
  - AIXTHREAD_MUTEX_FAST = ON
  - AIXTHREAD_SCOPE = S
  - SPINLOOPTIME = 4000
  - YIELDLOOPTIME = 20
- Enabled Servlet Cache
- Java version change from 1.6 to 8
- Disable AutoStart of following MDM Virtual Applications
  - MDM-native_E001
  - MDM-web-services-virtual-E001

10. MDM MQ Adapter Performance Tuning Guidelines

Here are the recommended settings for the MDM MQ Adapter

For incoming messages in MDM, "MDMMessaging_Request_ActivationSpec" is used. The thread pool, "WMQJCAResourceAdapter," is used to process inbound messages to MDBs. So, to control concurrency, another parallel workload on other activation specs using the same thread pool

- MaxServerSessions should be greater than desired concurrency Activation specifications >MDMMessaging_Request_ActivationSpec >Advanced properties >MaxServerSessions
- MaxPoolDepth should be higher than MaxServerSessions (a few extra) Activation specifications >MDMMessaging_Request_ActivationSpec >Custom properties >maxPoolDepth
- The (maximum) threadPool size for WMQJCAResourceAdapter should be higher than MaxPoolDepth. In addition, you would need to consider another parallel workload on other activation specs using the same threadPool. Servers > MyServerName >Thread pools >WMQJCAResourceAdapter >MaximumSize

- Set the “Maximum connections” for sessionPools for connection factories. Please set it to a value higher than threadPool size Resources -> JMS -> Queue connection factories >MDM Messaging Request QCF >Session pools Resources -> JMS -> Queue connection factories MDM Messaging Response QCF >Session pools
- Leave the “Maximum connections” for connectionPool as default (10) Resources -> JMS -> Queue connection factories >MDM Messaging Request QCF >Connection pool Resources -> JMS -> Queue connection factories >MDM Messaging Response QCF >Connection pool

11. Conclusion

In conclusion, organizations can benefit from improved transaction throughput, minimized system conflicts, and smoother data management experience. By implementing these recommendations, organizations can achieve a more efficient and resilient MDM environment capable of effectively handling current and future data management challenges.

References

[1] What's new in version 11.6 - IBM Documentation

[2] Extension and addition development planning - IBM Documentation

[3] Configuring the batch processor - IBM Documentation

[4] Matching engine options - IBM Documentation

[5] Understanding merge party survivorship rules - IBM Documentation

[6] Physical MDM configuration - IBM Documentation

[7] Infospheres MDM Probabilistic Matching Engine matching algorithm customization - IBM Documentation

[8] Creating the data source - IBM Documentation

[9] Setting Db2 system parameters - IBM Documentation

[10] Environment variables for performance tuning - IBM Documentation