WebSphere. software

IBM

Redpaper

Adam Muise

WebSphere Customer Center: Understanding Performance

This IBM® Redpapers™ publication provides insight into IBM WebSphere® Customer Center (WCC) transaction performance and scalability. It explores some of the most important features and implementation factors with regard to their performance impact. It also provides specific tuning recommendations for deployment.

Note: This paper is more relevant for WebSphere Customer Center version 6.5 and 7.0 and is intended for those who are experienced with WCC. Certain performance considerations and recommendations are also applicable to later releases of the IBM Master Data Management (MDM) products, including later releases of the MDM server.

Introduction

WebSphere Customer Center is IBM's solution to customer data integration and is a cornerstone in the IBM Enterprise Master Data Management solution. WCC is a flexible and highly scalable solution for any size business and is certain to be a key component of your business information hub. With this in mind, consider performance and scalability targets for such a critical system well in advance of deployment.

This paper provides a technical context to help you plan your WCC performance goals. It is a companion to the existing body of WCC documentation and resource expertise (see "Related publications" on page 12). It is not a substitute for WebSphere, DB2, or other related product expertise.

Performance considerations

In the first part of this paper, we provide a description of the most important performance related considerations for tuning the WebSphere Customer Center execution environment.

Transaction workloads

Identifying your predicted transaction workload mixes for average, peak, and seasonal changes is essential for planning performance in a future implementation or optimizing an existing deployment. The more you know about your workload, the easier it is to optimize. When making a table of the transactions in your various workloads, it is important to consider the following dimensions for each transaction.

In many cases, your transaction workload mix will differ significantly because of different usage patterns (for example, holidays, different times of the day, month-end processing, and so on). Therefore, it is useful to create a separate table for each of these variations. Consider the following dimensions when creating your transaction workload tables:

- Expected percentage of total workload

  This percentage is the predicted average percentage of the total workload.

- Exceptional deltas

  These deltas are the exceptional changes, especially peaks that might occur with regard to the transaction rate.

- WCC business objects involved in a transaction

  This list of business objects describes the various ones that are queried or modified in the transaction. The list aids in the understanding of transactional data access and the overall transaction choreography. If known, the cardinality of the business objects is important. For example, many customers in your queries might have two addresses (for example, TCRMPartyAddressBObj), but you can only exercise one address for updates.

- Inquiry level of transaction (where appropriate)

  For many WCC services that perform queries, an inquiry level is used by WCC as a directive to determine the granularity of the response. For example, when retrieving data about a customer’s contracts, it might not be necessary to retrieve all of the related contract components that can be associated with their contracts. Therefore, an inquiry level is specified that modifies the scope of the query. Reducing or increasing the required query level has an obvious impact on the amount of work that the WCC query must perform.

- Acceptable response time threshold

  This threshold is the maximum value that is tolerated for the average transaction response time. This value should be derived from either a service level agreement (SLA) or business expectations.

While mapping out transactional workloads might seem onerous, the value it can provide for helping to create expectations on database access patterns and pinpoint performance problems is immeasurable. The exercise of mapping out the WCC business objects involved might point to redundancies that can be introduced when combining multiple WCC services together.

For example, if your organization is consuming one of existing WCC “get” transactions with an inquiry level that returns an address object that you do not require, you can easily identify and correct the oversight, eliminating redundant queries in the transaction. In an early WCC planning scenario, this information can help model expectations for performance response times. Finally, having a detailed map of your transaction workload is an excellent and often required reference for optimization and maintenance efforts.

Planning for data

Obviously, the most important step in any information hub is planning the management of information. As a general starting point, consider the following dimensions for each WCC table when accounting for the performance impact of your data profile:

- Row count

  The actual quantity of data in the table helps you to plan your tablespace size and understand your potential hot database queries.

- Predicted row count growth rate

  Growth rates are primarily important for effective tablespace planning, storage allocation, and ongoing table maintenance. An early understanding of how your data model will grow also helps you to evaluate your approach to position your data into the WCC data model or plan for data model extensions. Scalability issues can often be avoided by such proactive work.

- Statistical distribution of data

  For the purposes of planning table indexes, it helps to understand the frequency of duplication in table columns that can be searched. An example of this might be a name column where there is frequent duplication of both first and last names. Therefore, the efficiency and performance of the index is degraded. While core WCC tables are already optimized for most services, it is best to be aware of potential problems that can arise with the customization to WCC transactions, extensions, or queries that are common to many implementations.

- Frequency of access

  The frequency of access helps to highlight obvious hot spots. While this might be a broad generalization when considering the complicated queries of transactions, a careful consideration of the magnitude, frequency, and repetition of data access helps you understand how to ultimately size the database buffer pool.

- History or additional database triggers

  Triggers typically have a low overhead, but it is important to remain aware of them because, regardless, they are part of the transaction response time. Furthermore, understanding WCC trigger activity and the frequency of table access helps you to plan for the growth of your WCC history tables.

Although hundreds of tables are in the basic WCC model, in most implementations, about 15 main tables require careful consideration because of the frequency of their access and relative size. These tables form critical performance hot spots because they and their respective database indexes are accessed concurrently and frequently. Therefore, it is important to identify them before implementation and optimization.

After the tables are identified, you can use the following common optimization strategies to help achieve your performance goals and ensure that your WCC database scales appropriately now and in the future:

- Tablespace planning

  By taking into account the row count, predicted growth rate, and frequency of access dimensions of your tables, you can divide the high activity tables evenly among your table spaces. This “divide-and-conquer” strategy allows for more specific tuning of table spaces and buffer pools based on the their access pattern. Optimized table spaces and buffer pools improve query heavy workloads that often make repeated visits to the same data.

- Table partitioning

  Table partitioning is another divide-and-conquer strategy that can help alleviate overhead on the database during highly concurrent workloads. This strategy is most applicable to those implementations on DB2 on z/OS®. Typically, the partitions are created based on an even distribution of the WCC primary keys. The default behavior of WCC is to generate keys based on a date and a random number. However, primary key generation is a pluggable component and can be substituted to accommodate a partitioning strategy if considered early.

  For example, you might want to have separate key generators for different geographical or functional divisions of your organization, thus enforcing a more predictable workload across your various tables. Table partitioning is a complex decision that involves storage considerations in addition to performance factors. Discuss such a strategy in detail with your database administrators and solution architects.

- Identifying exceptional cases

  There are exceptions to the norms for every data model, often a small subset of records that have dramatically different relationships and access patterns. In WCC, this can be customers who have an unusual amount of related business objects or simply customer records that are accessed in great concurrency and frequency. In these diverse and exceptional cases, we cannot provide a concrete solution to the issue. Rather it is suggested that you identify and plan early for exceptional cases.

Suspect duplicate processing

One of the most valuable features in WCC is its ability to identify and prevent duplicate party information automatically. The additional query workload to support this feature adds some overhead. Therefore, planners must be aware of what to expect and how to mitigate any response time and throughput challenges. With WCC, consider the performance implications of *suspect duplicate processing* for both the initial population of the WCC database and runtime workloads.

Estimating the performance overhead of suspect duplicate processing on your transaction workload is complicated and depends greatly on the following factors:

- Data quality analysis

  Data quality analysis is key to planning and predicting suspect duplicate processing workload and overhead. Statistical analysis of the input data is key to determining what critical data changes must be defined, what suspect searches will yield the best data quality for matching, and ultimately what to expect in terms of performance overhead. Frequently, IBM InfoSphere™ Information Analyzer is used to assess data quality and matching considerations.

- Critical data changes in your transactional workload

  Since suspect duplicate processing operates only on a party’s critical data, workloads that contain many additions or modifications to this critical data are more impacted than those with fewer modifications. Query (read-only) WCC transactions are not impacted directly by suspect duplicate processing. Some examples of critical data are the party’s name, address, and tax identification number.

- Suspect candidate pool

  If a party that is added or modified has many potential suspects, the work that suspect duplicate processing must do increases. In the default configuration of WCC, many transactions that involve critical data changes or additions return any potential suspects based on the defined search criteria. The performance impact is less significant in situations where a definite suspect is found, because the default behavior of WCC is to avoid the actual addition or modification. However, in cases where the input data quality is low, many suspects candidates can be found by the suspect searches that might not eventually become matches. In cases where there are excessive suspect candidates, optimizing the searches or lowering the global WCC search limit is recommended.

- Suspect candidate party retrieval

  Most of the overhead in suspect duplicate processing is because of the “getParty” service calls for each of the parties in the suspect candidate pool. The easiest way to reduce this overhead is to reduce the number of suspect candidates that are found to only those that are legitimate candidates, as mentioned previously. The overhead of the “getParty” service calls can be reduced through customization of the inquiry level or by reducing the subcomponent calls to only what is required by the match rules.

- Matching engine

  The default WCC matching engine is internal and deterministic. A detailed guide to suspect scoring for this matching engine is in the *WebSphere Customer Center Developer’s Guide*.

  WCC documentation: The following documentation is available only after you purchase the WebSphere Customer Center product:

  - *WebSphere Customer Center Developer’s Guide*
  - *WebSphere Customer Center Understanding and Planning Guide*
  - *WebSphere Customer Center System Administration Guide*
  - *WebSphere Customer Center Transaction Reference Guide*

  Another option is to pair WCC with the IBM InfoSphere QualityStage for a probabilistic matching engine as well as name and address standardization. Because the WCC application makes external remote method invocation (RMI) calls to the QualityStage server, this solution is slower than the built-in matching engine, although it provides significant additional functionality to suspect duplicate processing. Alternatively, you can customize your own matching engine. Naturally the degree of performance overhead is determined by the efficiency of your own code.

- Database factors

  Other than the quality of data of the requests, there are a few database specific considerations. The most important consideration is the size of the database. While it has been shown that WCC implementations employing DB2 scale well, there can be an increase in response time where more suspects must be processed. It is also worth noting that suspect duplicate processing significantly increases query workload on the WCC tables with critical data.

From data collected internally, we made the following observations about the performance impacts of suspect duplicate processing performance:

- Average transaction response increases only for those transactions that add or modify critical data. This increase can be up to 40% for transactions employing all of the critical data used in WCC.
- As the average transaction response time increases, the transactions per second (TPS) drops proportionately. The drop in TPS only applies to those transactions that involve suspect duplicate processing. No bottlenecks are introduced provided that your infrastructure can handle the extra overhead on the application and database server.

- Suspect duplicate processing causes an increase in the overall CPU usage. The increase occurs at both the database and application layer due to the greater number of queries. The increase in CPU is typically proportional to the change in TPS.
- When loading your data into WCC for the first time, or running large batches of customer adds and updates, it is important to execute DB2 statistics (the `runstats` command) on your critical data tables progressively as the load progresses. If your database is empty, a good rule of thumb is to execute the `runstats` command after loading only 10,000 parties (persons or organizations). After the first 10,000 parties, you can increase the interval significantly. The interval for statistics collection depends on how much you expect your data distribution to change as you add parties. Watching for significant changes in load throughput might indicate that it is time to re-execute the `runstats` command.
- Because of the increase in database queries, suspect duplicate processing workloads benefit immensely from properly sized buffer pools.
- In unusual circumstances where there is a high degree of concurrent access and modification of a few specific records, suspect duplicate processing can increase the response time well above the typical estimated overhead due to increased contention over hot database records. This unusual circumstance can occur when one person or organization is frequently updated and queried by many different WCC services, including suspect duplicate processing searches.

For more information about WCC suspect duplicate processing, see the *WebSphere Customer Center Developer's Guide* and the *WebSphere Customer Center Understanding and Planning Guide*.

Transaction Audit Information Logging

By using WCC, you can account for additions and modifications that are made to parties with the Transaction Audit Information Logging (TAIL) service. In order for the service to provide this information on demand, any WCC services that add or modify data will have an additional overhead when TAIL is enabled.

Consider the following points regarding performance when you enable TAIL:

- Complex WCC transactions typically contain other services. TAIL keeps an internal log of these child services for auditing purposes. Therefore, the more business objects you modify or change, the more overhead you are likely to experience.
- The WCC services that retrieve data from the TAIL tables are often low in a transactional workload and are not typically a performance concern. The only concern is the WCC services and those transactions that modify or add data because they will trigger TAIL.
- TAIL overhead can add up to 45% to response times of add and update transactions, depending on the amount of WCC business objects being modified.
- TAIL overhead is the same for both add and update transactions.
- TAIL decreases TPS in those affected transactions in proportion to the overhead it adds. TAIL does not introduce any bottlenecks that might impact other transactions in the workload.
- TAIL increases usage on the WCC database tables of TRANSACTIONLOG, INTERNALLOG, INTERNALLOGTXNKEY, and INTERNALTXNKEY.

For more information about TAIL in WCC, see the *WebSphere Customer Center Developer's Guide* and the *WebSphere Customer Center Understanding and Planning Guide*.

Smart inquires

Smart inquiries is a WCC feature that allows an implementer to turn off particular sets of WCC data model queries that are not in use. This simple change avoids the execution of extraneous queries, significantly reducing transaction response time and increasing TPS in workloads in almost all transaction workload scenarios. Since this feature can modify the functionality of the product and your use of the WCC data model can evolve over time, use careful consideration when implementing this feature.

Consider the following points for using smart inquiries:

- Because most WCC services include embedded inquiries, this feature often has benefits in all kinds of WCC services.
- The performance benefit is contingent on how many of your transactions are querying the unused portions of the WCC data model, how often they are queried, and how many unused portions you turn off.
- This feature reduces response time and increase TPS in those affected transactions.
- Optimizations typically result in a significant savings in CPU usage on the application and database layers, as well as an overall reduction in WCC database workload that is proportional to the amount of queries reduced.

Summary data indicators

Another feature that is related to smart inquiries avoids extraneous database requests. This feature is called *summary data indicators*. Where the smart inquires feature side-steps unnecessary queries for all WCC services, the summary data indicators feature avoids queries on a per-transaction basis. When the summary data indicators feature is enabled, WCC business objects with child entities are tracked in the CONTSUMMARY database table. For every applicable business object in a query, this table is consulted to determine if it is necessary for WCC to make an additional query for child objects.

For example, a WCC query for a person without an identifier can skip the database query and application overhead if no identifier is present for that particular party. If the same query is executed for a person with an identifier present, then WCC proceeds to retrieve it as normal.

When using summary data indicators, keep in mind the following considerations:

- The CONTSUMMARY table must be maintained for all add and update transactions that add or remove child objects, adding slightly more overhead to these transactions.
- The CONTSUMMARY table has a large record count and must be considered in your “hot” database table list because many queries make requests to it. Because the index is unique (based on the contact id column, cont_id) and most of the column values are of the type SMALLINT, these are relatively low-overhead SELECT statements.
- This optimization applies best for WCC databases that have a significant number of parties without particular child objects. If all or most of your parties possess a particular child object, then it does not make sense to employ this feature. If you did not intend to use a particular child object in your implementation, then use smart inquiries to turn off that portion of the model.

Figure 1 shows the relation of the performance factors to the response time and TPS value.

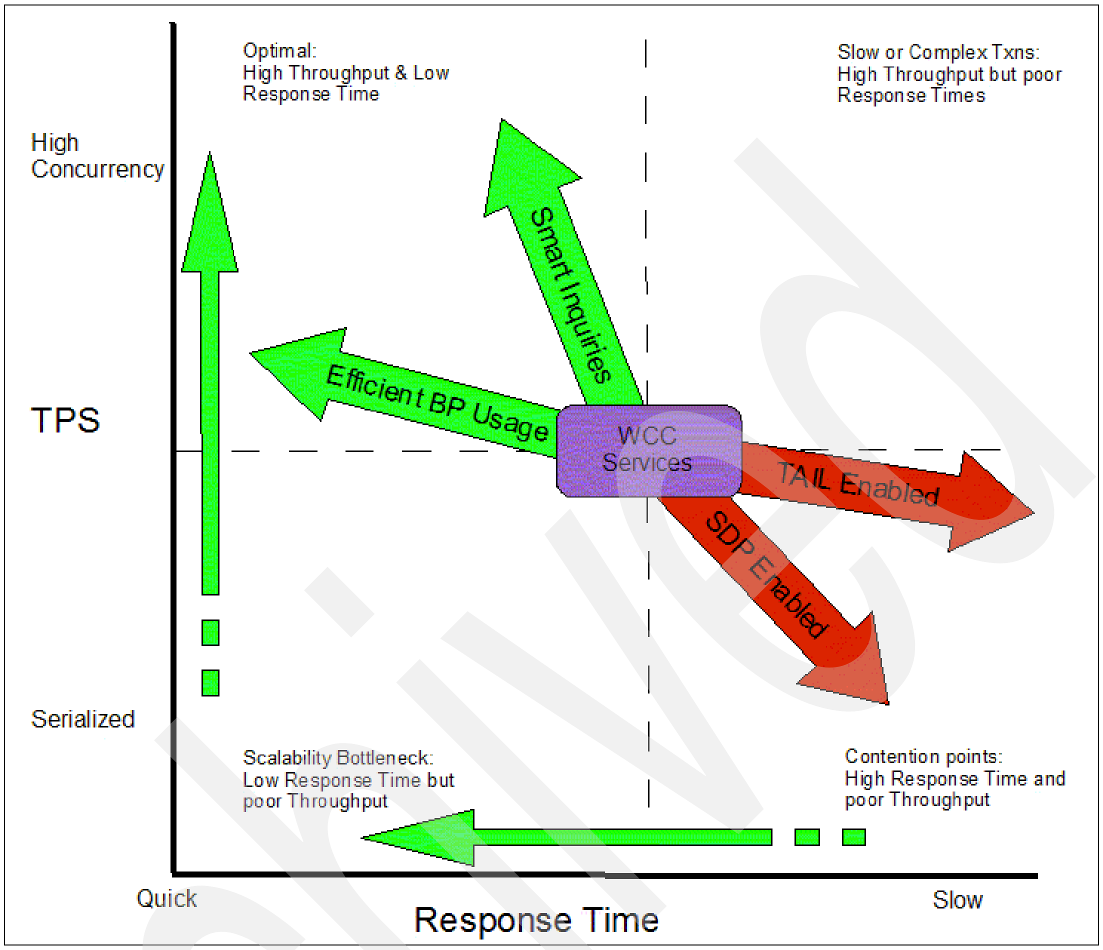

Figure 1 Relation of performance factors

Diagnostic and performance tuning recommendations

In this section, we provide practical recommendations for tuning the WebSphere Customer Center components.

Optimizing WCC transactions

The best time to consider transaction performance is when translating your functional requirements into WCC services. At this point, you can eliminate redundancies, decide on customized transactions, and even predict potential performance hot points.

Usually one or more of the hundreds of WCC services can address your functional needs. However, some situations might require a custom transaction or a composite transaction of multiple WCC services. For example, if you need to retrieve simple party information and detailed information about a particular contract and used “getFSParty” with the highest inquiry level, you might retrieve all of the contract, address, and contact information for that party. Instead, by choosing more fine-grained services, such as “getFSParty” with a lower inquiry level followed by a “getContract”, you might get exactly the information you need without the additional calls.

Making more fine grained service calls has numerous benefits on performance. It is especially beneficial for reducing database locking, Java™ virtual memory (JVM™), and application CPU usage. By breaking larger transactions into smaller transactions, where functionally appropriate, less time is spent holding onto resources in the application and database layer, resulting in higher concurrency and better performance.

Understanding the details of the WCC services helps to make these decisions. Consult the *WebSphere Customer Center Transaction Reference Guide* for detailed information about each WCC service and their child components.

WCC tuning

Most of the optimization for WCC can be accomplished by taking the previously mentioned steps of planning transactions carefully. However, we make the following WCC-specific configuration recommendations:

- Disable suspect duplicate processing for exceptional workload cases that might not benefit from it. This is especially recommended for initial WCC data migration or loading because not all of the customer data is present for the suspect duplicate processing to make an accurate suspect list during most of the data loading process. Instead, use the *evergreening process* after the data is loaded. Consult the *WebSphere Customer Center Understanding and Planning Guide* for more information about evergreening data.
- Disable TAIL if you do not require that level of auditing in your implementation. In most cases, also disable TAIL for initial WCC data migration or loading.
- Use the smart inquiries feature whenever possible.
- Disable WCC performance tracking for regular production workloads. Instead consider using the services activity monitoring feature of WCC (available in WCC version 7.0 or later).
- Reduce WCC logging levels to the minimum required for your infrastructure.
- Use the search exclusion feature (available in WCC version 7.0 or later) where appropriate to avoid producing excessively large results.

WebSphere Application Server tuning

Optimal WebSphere Application Server settings vary from implementation to implementation. However, always address the following items for each WebSphere Application Server instance:

- Ensure that the Object Request Broker (ORB) thread pool is equal to the maximum amount of concurrent RMI (direct calls to the WCC ServiceController stateless session bean) WCC transactions expected for each WCC instance. ORB threads block WCC service requests. Therefore, the maximum thread pool limit is your theoretical maximum number of concurrent users.
- Ensure that the Web container thread pool is equal or tuned to a high enough level to accommodate Web service requests, if applicable for your application. In most cases, this thread pool does not need to be adjusted because Web container threads are non-blocking and can handle multiple WCC service requests concurrently.
- Set the Enterprise JavaBeans™ (EJB™) cache size to 3500, which is the optimal setting for WCC.

- On the Java Database Connectivity (JDBC™) data source used for WCC (default Java Naming and Directory Interface (JNDI) name is `jdbc/DWLCustomer`), set the Prepared Statement cache size to 300, which can improve transaction response time by 5% to 15%.

  Attention: Setting the cache to higher values can cause a negative performance impact.

- Use the default logging level (or lower) that is set by WebSphere.

For more information about WebSphere Application Server tuning, see the WebSphere Application Server 6.0 Information Center and the IBM Redbooks® publication *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

DB2 tuning for Linux and AIX

Tuning DB2 for WCC is similar to tuning it for other Online Transaction Processing (OLTP) workloads. Much of the details are covered in far greater detail in DB2 documentation. Keep in mind the following high level suggestions and WCC-specific considerations for tuning DB2:

- Recommend a 64-bit operating system and 64-bit DB2 implementation. The client libraries are not important.
- Allocate sufficient memory to DB2.
- Ensure that you have large enough buffer pools balanced appropriately based on your tablespace activities.
- Using single buffer pool per page size is typically the most effective approach. Using multiple buffer pools requires more knowledge on different access patterns and data volumes of various tables to ensure proper allocation of memory among all the buffer pools. When trying to use multiple buffer pools, a lack of such knowledge or misunderstanding available information can lead to a waste of memory and poor performance.
- Minimize I/O constraints and fully use the capacity of your I/O system by spreading the table spaces evenly across all the available disks (physical spindles).
- Use a separate set of disks for the file system that is used by the DB2 transaction log files. The location of the log files is specified by the “Path to log files” database configuration parameter.
- Use JDBC type 4 drivers for all WCC data sources that use DB2.

  Note: Starting with WCC version 7.x, type 4 is the default.

For more information about DB2 tuning, see the article “DB2 Tuning Tips for OLTP Applications” and the other DB2 tuning articles listed in “Related publications” on page 12.

IBM JVM tuning

Due to the data-intensive nature of WCC, optimizing the IBM JVM can eliminate bottlenecks and improve transaction throughput. Run your WCC application instance in a separate JVM from other applications in your environment.

For JVM tuning, keep in mind the following considerations:

- The Java heap size of a resting WCC application deployed in WebSphere version 6.0 is approximately 110 MB. To accommodate peaks and valleys, set your minimum heap size to 512 MB and your maximum heap size to 1024 MB. These settings give the optimal balance of headroom for types of workloads without inducing heap fragmentation. If your heap size must be increased, add another WCC instance and balance the workload. Going beyond the 1024 MB heap size is acceptable. However, consider adding another WCC instance at that point to maintain the optimal conditions for JVM garbage collector.
- If your workload contains long-running transactions that call multiple WCC services in the context of a single Java Transaction API (JTA) transaction, then objects held cannot be collected as quickly as with smaller transactions. The result is that your average heap usage will grow, your garbage collection cycles will be more costly, and consequently your throughput will be degraded. If your workload contains many of these types of transactions and you cannot reduce their complexity, plan to use more JVMs to accommodate the memory usage.
- In most situations, the default garbage collection policy is optimal. However, in cases where the workload is primarily composed of larger and more complex transactions, the “subpool” garbage collection policy frequently yields better throughput. Large WCC transactions can be memory intensive (in excess of 3 MB per average WCC request). This policy optimizes the memory allocation process, reducing contention when WCC needs to create more objects to accommodate the larger transactions.

  Subpool garbage collection policy: The subpool garbage collection policy is available only on AIX® implementations of IBM Java and can be invoked by adding the following directive to your JVM:

  ```
  -Xgcpolicy:subpool
  ```

- It is helpful to profile your memory usage under all of your defined transaction workloads to understand where memory can become an issue. The easiest way to do this is to enable verbose garbage collection (a check box in the WebSphere Administrative Console or a command line directive of `-verbosegc`), which logs all garbage collection events to the `native_stderr.log` file by default. You can use one of the many free tools available from IBM to analyze the output of the verbose garbage collection, such as the IBM Pattern Modeling and Analysis Tool for Java Garbage Collector.

For more information about IBM JVM, garbage collection, and the Java Memory Model, see the *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2 Diagnostics Guide* at the following address:

http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/diagnosis/diag142.pdf

WCC performance tracking

WCC provides service level performance tracking that can help you gain insight into where your WCC transactions are spending their time. This is extremely helpful as a starting point for performance troubleshooting or optimization. If one of your transactions has a poor average response time, then you can obtain a high level picture of where the service is spending most of its time at the WCC component level.

For example, you might find that an updatePartyAddress transaction is spending a significant portion of the response time in a customized address standardization that was added to the transaction workflow. With WCC performance tracking, you can begin to focus your efforts in the right location.

Restriction: In general, do not use WCC performance tracking on a production system. However, if you must use this feature temporarily to troubleshoot a performance issue, keep it at Level 1 to reduce unnecessary overhead.

For details about how to enable and use WCC performance tracking, see the *WebSphere Customer Center Understanding and Planning Guide*.

Related publications

Several sources are available for documentation and more information.

The following documentation is only available after you have purchased the WebSphere Customer Center product:

- *WebSphere Customer Center Developer's Guide*
- *WebSphere Customer Center Understanding and Planning Guide*
- *WebSphere Customer Center System Administration Guide*
- *WebSphere Customer Center Transaction Reference Guide*

For WebSphere documentation, consult the following sources:

- WebSphere Application Server Information Center

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/welcome_base.html

- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392

  http://www.redbooks.ibm.com/abstracts/sg246392.html?Open

For DB2 documentation, consult the following sources:

- *Automatic Configuration for IBM DB2 Universal Database*, REDP-0441

  http://www.redbooks.ibm.com/abstracts/redp0441.html?Open

- *DB2 UDB V8 and WebSphere V5 Performance Tuning and Operations Guid*e, SG24-7068

  http://www.redbooks.ibm.com/abstracts/sg247068.html?Open

- "Improve database performance on file system containers in IBM DB2® UDB V8.2 using Concurrent I/O on AIX" on IBM developerWorks®

  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0408lee/

- "DB2 Tuning Tips for OLTP Applications" on developerWorks

  http://www-128.ibm.com/developerworks/db2/library/techarticle/anshum/0107anshum.html

- "Top 10 performance tips" on developerWorks

  http://www.ibm.com/developerworks/db2/library/techarticle/hayes/0102_hayes.html

- "DB2 performance testing and monitoring using Rational® Performance Tester, the best practices and rules of thumb" on developerWorks

  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0801liu/

- IBM DB2 Information Center

  http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

The team that wrote this IBM Redpaper

This paper was written by Adam Muise, Information Management Performance Evangelist, for IBM Software Group in Ontario, Canada. You can contact Adam Muise by sending e-mail to amuise@ca.ibm.com, or his manager, Yongli An, by sending e-mail to yongli@ca.ibm.com.

Thanks to the following teams for their contributions to this project:

- The global Master Data Management (MDM) performance team
- MDM development team
- The IBM Java/JIT performance team

Notices

This document REDP-4499-00 was created or updated on May 29, 2009.

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an e-mail to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Redpaper™

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

AIX®	IBM®	Redpapers™
DB2 Universal Database™	InfoSphere™	Redbooks (logo) ®
DB2®	Rational®	WebSphere®
developerWorks®	Redbooks®	z/OS®

The following terms are trademarks of other companies:

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

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