Examining findings

This section includes the following topics:

How to identify performance problems

In most cases, a Precise for .NET workspace displays information in the context of a specific instance and time frame. However, if you want to view findings for another time frame or instance, you can change these settings using the respective dropdown lists.

How to investigate findings

When you start investigating the findings, it is good practice to start with the findings that have the highest severity rankings in the Findings table.

To investigate a finding

1. Identify the findings with the highest severity rankings (red, orange, and yellow icons where red is the highest severity and yellow the lowest) in the Findings table on the Dashboard tab.
2. Select the finding you want to investigate.
3. Read the information under the Highlights tab to understand what the problem is about.
4. After you have studied the information provided, select the Advice tab and perform the recommendation(s) that best suit(s) your needs. If you need additional information, read the information available under the Background tab.

Heavy Entry Point

Heavy entry points may indicate a potential performance irregularity in the entry point's underlying call path. When viewing information for "All Instances" in the invocation tree, a high entry point finding across multiple instances will clarify if the irregularity is a result of the entry point or if it is a result of a specific instance.

The finding is based on the total service time values, whereas by default, the invocation tree is displayed according to the average service time. As a result, the specified heavy entry point is not necessarily found towards the top of the invocation tree.

Working with the finding

To effectively locate the root cause of the performance finding, perform the following:

- Select the link in the expanded finding area. This will refresh the Invocation tree and Analysis tabs to focus on the entry point, which will now be the selected node in the Invocation tree. In the new Highlights tab, examine the entry point's performance overtime activity and follow the specific findings to easily navigate to and analyze the root cause of the performance issue.

Frequent SLA Breaches

SLA thresholds are defined to help the user pinpoint HTTP entry points experiencing performance issues according to specific criteria. Frequent SLA breaches and near breaches can be caused by an underlying performance issue.

Working with the finding

To effectively locate the root cause of the performance finding, perform one of the following:

- Click on the entry point's link, and then look at the overtime SLA behavior to locate and zoom in to a specific (problematic) time frame. View the findings for that time frame and drill down until you locate the root cause.
- Select the root level of the invocation tree and select the Entry Points tab. A high rate of SLA breaches across the application could result from overall resource exhaustion. Open the Memory and Statistics workspace to learn more about the environment performance issues, such as high memory usage, CPU usage and so on.

- Go to **AdminPoint>Settings>SLA** to view the threshold definitions. (When you have too many SLA breaches, it may be a result of thresholds that are not defined appropriately for your application).

Heavy Method Contribution

A high work time for a specific method (reflecting the method's work time only, without the underlying call path), can indicate a performance issue within the context of that method.

In the same way, a high work time for a specific occurrence of a method invoked multiple times in the Invocation tree can indicate a performance issue within the context of that specific occurrence.

Working with the finding

To effectively locate the root cause of the performance finding, perform the following:

- Examine the heaviest occurrences further by following the featured link.

The invocation tree opens to the method's heaviest call path, facilitating effective navigation to the root cause. Examine the information displayed and look at the overtime graph and findings to drill down further to find the root cause of the performance issue.

> By default, information is displayed for the heaviest method's call path. To investigate the other call paths, select them from the invocation tree. (They are highlighted in bold).

Excessive Lock Time

A significant percentage of the selected entity's total service time is spent waiting for lock acquisitions.

Waiting for locks means that the online activity (or batch process) is on hold, and is waiting for another activity or process to release the lock. Typically, eliminating locks can improve the performance of your transactions.

A possible solution is to consider tuning your transaction performance to reduce the time spent waiting for locks.

Slow DB Request Execution

A significant percentage of the selected entity's total service time is spent waiting for a specific DB request execution. A possible solution is to tune your DB requests to optimize transaction performance.

Working with the finding

To effectively locate the root cause of the performance finding, perform one of the following:

- Click on the queries to find their occurrences in the call tree, find the target DBs, and inspect their behavior over time.
- Examine the DB request further by following the featured link.

The SQL and Exit Points tab opens, displaying details for SQL statements that the selected entity or its underlying call tree is waiting for, according to their contribution to the overall performance. From this view of the overall external activity:

- Follow one of the slowest DB requests to drill-down within the call tree and check the load balancing information further by opening the Load Balance tab. Examine whether it is relatively heavier when being called from specific CLRs.
- If the database instance is monitored by Precise, follow the "Analyze" link in the Highlights tab to drill down to the respective expert view (for example, Precise for Oracle), and see how it could be more efficiently tuned.
- Consider parallelizing the external activity to run while processing is being done on the CLR side.

Slow Web Service Execution

A significant percentage of the selected entity's total service time is spent waiting for a specific external Web service invocation.

A possible solution is to tune the external Web service to optimize transaction performance.

Working with the finding

To effectively locate the root cause of the performance finding, perform one of the following:

- Follow the featured link and examine which call paths invoke the external Web service and which call paths are the heaviest. From this view of the overall external activity:
    - Follow one of the web service requests to drill-down within the call tree and check the load balancing information further by opening the Load Balance tab. Examine whether it is relatively heavier when being called from specific CLRs.
- If the Web service is running on a .NET instance which is monitored by Precise, follow the "Analyze" link in the Highlights tab to drill down to the respective expert view (for example, Precise for Oracle), and see how it could be more efficiently tuned.
- Consider parallelizing the external activity to run while processing is being done on the CLR side.

- Check the load balancing information of the Web service by opening the Load Balance tab. Examine whether it is relatively heavier when being called from specific CLRs.

Heavy Exit Point

A significant percentage of the selected entity's total service time is spent waiting for external activity.

A possible solution is to consider tuning your transaction performance and the time spent executing external activity.

Working with the finding

To effectively locate the root cause of the performance finding, perform the following:

- Examine the external activity further by following the featured link.

The SQL and Exit Points tab opens, displaying details for exit points that the selected entity or its underlying call tree is waiting for, according to their contribution to the overall performance. DB activity is featured in this list as SQL statements.

- Click on a link to examine the call paths that invoke the external activity, and focus on the heaviest.
- If the exit point is monitored by Precise, follow the "Analyze" link in the Highlights tab to drill down to the respective expert view (for example, Precise for Oracle), and see how it could be more efficiently tuned.
- To understand the influence the exit point has on the entire application, open the Impact tab to view the impact of the exit point on all entry points and call paths in the application.
- Check the load balancing information of the exit points by opening the Load Balance tab. Examine whether it is relatively heavier when being called from specific CLRs.
- Consider parallelizing the external activity to run while processing is being done on the CLR side.

Significant ADO.NET Activity

A significant percentage of the selected entity's total service time is spent waiting for DB activity invoked by a JDBC request.

This can result from a specific heavy statement or a collection of many, relatively short statements, being executed. A possible solution is to consider tuning your transaction performance to reduce the time spent executing DB statements.

Working with the finding

To effectively locate the root cause of the performance finding, perform one of the following:

- Examine the external activity further by following the featured link.

The SQL and Exit Points tab opens, displaying details for exit points that the selected entity or its underlying call tree is waiting for, according to their contribution to the overall performance. DB activity is featured in this list as SQL statements. From this view of the overall external activity:

- Consider unifying queries to eliminate communication and query overheads.
- Follow one of the heaviest SQL statements' links to drill-down within the call tree to locate the target DB and perform one of the following:
- Check the overtime activity graph and summary area in the highlights tab.
- Check the load balancing information further by opening the Load Balance tab. Examine whether it is relatively heavier when being called from specific CLRs.
- Consider parallelizing the queries to run while processing is being done on the CLR side.
- To understand the influence the exit point has on the entire application, open the Impact tab to view the impact of the exit point on all entry points and call paths in the application.

Significant External Activity

A significant percentage of the selected entity's total service time is spent waiting for external activity.

This can either indicate that a specific external activity is experiencing performance issues, or a large number of external calls, each of them relatively short, producing a high total service time.

Working with the finding

To effectively locate the root cause of the performance finding, perform one of the following:

- Examine the external activity further by following the featured link.

The SQL and Exit Points tab opens, displaying details for exit points that the selected entity or its underlying call tree is waiting for, according to their contribution to the overall performance. From this view of the overall external activity:

- Consider unifying requests to eliminate communication overheads.
- Consider using an internal cache mechanism for reducing external calls, in case the same calls are being invoked repeatedly.
- Follow one of the heaviest exit points' links to drill-down within the call tree, and perform one of the following:
- Check the overtime activity graph and summary area in the Highlights tab.
- Check the load balancing information further by opening the Load Balance tab. Examine whether it is relatively heavier when being called from specific CLRs.

- To understand the influence the exit point has on the entire application, open the Impact tab to view the impact of the exit point on all entry points and call paths in the application.
- If the exit point is monitored by Precise, follow the "Analyze" link in the Highlights tab to drill down to the respective expert view (for example, Precise for Oracle), and see how it could be more efficiently tuned.
- Consider parallelizing the external activity to run while processing is being done on the CLR side.

Tuning Opportunities Detected

The selected method showed a high work time. This may indicate that the selected method is the last branch of the call tree, or that visibility to the selected method's underlying call path can be enhanced by adding instrumentation.

Working with the finding

To effectively locate the root cause of the performance finding, perform one of the following:

- If the information regarding the method and its performance metrics is sufficient, forward this information to the .NET expert or developer for next-level handling using either the email or print option.
- If more accurate pinpointing is needed, increase the level of visibility by adding instrumentation for all methods in the selected method's call tree by updating the instrumentation definitions. Following a CLR restart, you will see a detailed breakdown of the work time of the selected method and its call tree, enabling easy identification of the specific problematic method. For more information, see the About instrumenting all calls from a method section in the Precise Administration Guide.

Locks Detected

While executing the selected context and its underlying call tree, time was spent waiting for lock acquisitions.

While waiting for lock acquisition, the online activity (or batch process) is put on hold until another activity or process acquires the lock. Therefore, eliminating locks can improve the performance of your transactions.

Unbalanced Activity

The selected entity was invoked in multiple CLRs and encountered significantly different service times across the CLRs.

A transaction executed on a cluster of CLRs may encounter different service levels per execution, depending on the activity of the specific cluster's CLRs executing the transaction.

Working with the finding

To effectively locate the root cause of the performance finding, perform one of the following:

- Examine the affected paths and transactions by following the featured link.

The Load Balancing tab opens, displaying an overview of the selected entity's behavior across the different CLRs, and providing the ability to select the specific CLR and examine its relative performance against the application's average.

- Select a CLR row and check its overtime service time and number of invocations.
- Where there is a high or growing number of invocations over time, the difference in service time may be the result of load balancing issues. The affected CLRs may be getting more requests than their counterparts, and may not be able to cope with the increasing load. Check your load balancing component, as well as the scalability of the CLR's software.
- When there are no load balancing issues, but the CLR still has a high service time, this may be a result of a resources shortage on one or more of the CLRs. Go to the Memory & Statistics workspace and examine the behavior of the affected CLRs, to determine whether a resource issue is affecting the performance of the application running on it. Typically, in case of resource shortage, more than one entry point will be affected.
- When there are no load balancing issues and there is no resource shortage, select the slower CLRs from the Load Balancing tab, and drill down to examine the behavior of the entity on the specific CLR.

Impact on Multiple Entry Points

The selected method/SQL is invoked from multiple call paths and therefore affects the performance of the instance in a cumulative manner.

Therefore, improving the performance of the selected method/SQL will impact more than its current call path, and may improve the performance of additional paths and transactions containing the method.

Working with the finding

To effectively locate the root cause of the performance finding, perform the following:

- Examine the affected paths and transactions by following the featured link.

The Impact tab opens, displaying a list of entry points and call paths affected by the selected method. Tuning methods/SQLs invocations with notably high impact rates will positively affect the overall performance.

- Take note of high variations between method service times when called from different paths. Such variations may indicate a dependence of the method's performance on context. Therefore, drill down to the problematic context(s) for further tuning.

Excessive DB Connection Strings Detected

Your application may be experiencing bad connection string usage.

Too many different connection strings were used when connecting from your application to the DB.Each connection string in ADO.NET is using a specific pool. Many different connection strings cause bad pooling and connection opening overhead.

Working with the finding

To effectively locate the root cause of the performance finding, perform the following:

- Use the link to get more information regarding the Databases and the number of their connection strings.

Explicit Calls to Garbage Collection

During the selected time frame, your application performed calls to GC methods.

Calling GC methods from application code is incorrect, since .NET architecture was designed to handle all garbage collections by itself. Every interference with the GC method harms performance.

Working with the finding

To effectively locate the root cause of the performance finding, perform the following:

- Select the link in the expanded finding area. This will refresh the invocation tree and will find the explicit calls to the GC. Each call must be examined carefully and be omitted unless found crucial.