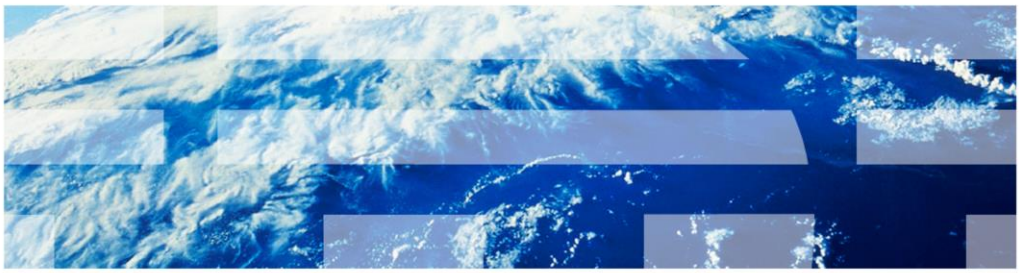
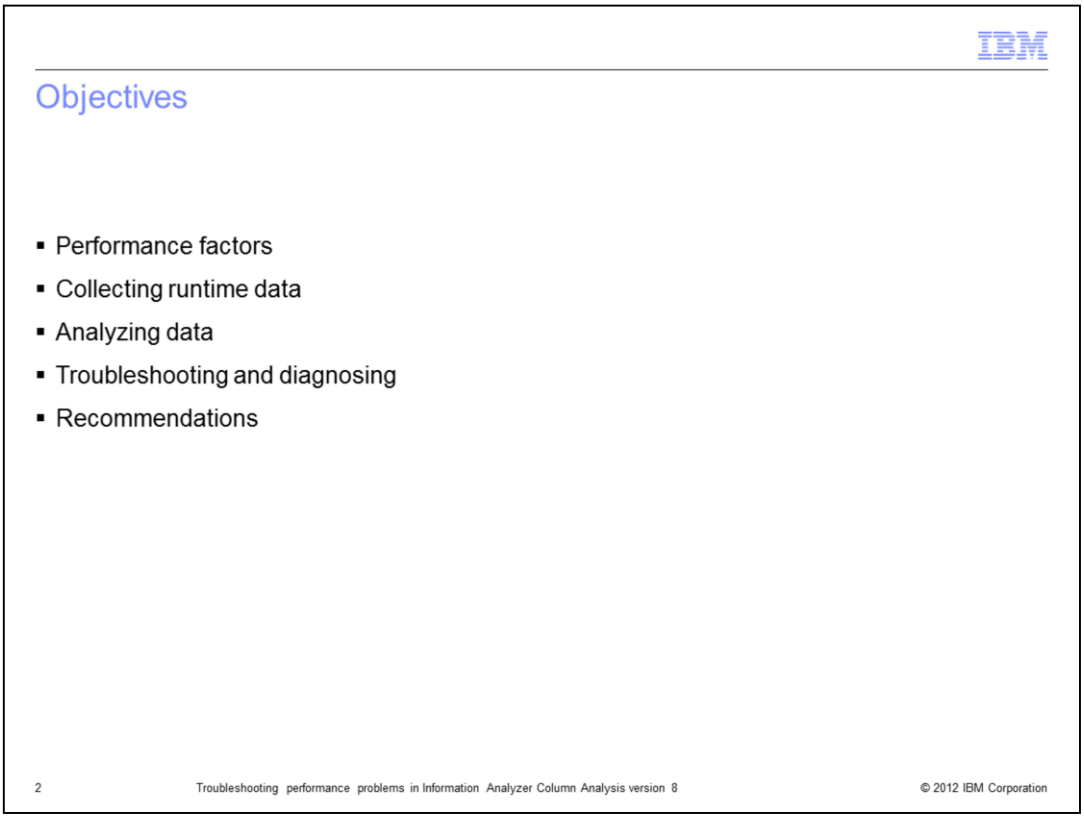
IBM

InfoSphere Information Server

Troubleshooting performance problems in Information Analyzer Column Analysis version 8



This presentation will explain how to troubleshoot performance problems in Information Analyzer Column Analysis for version 8 and higher.

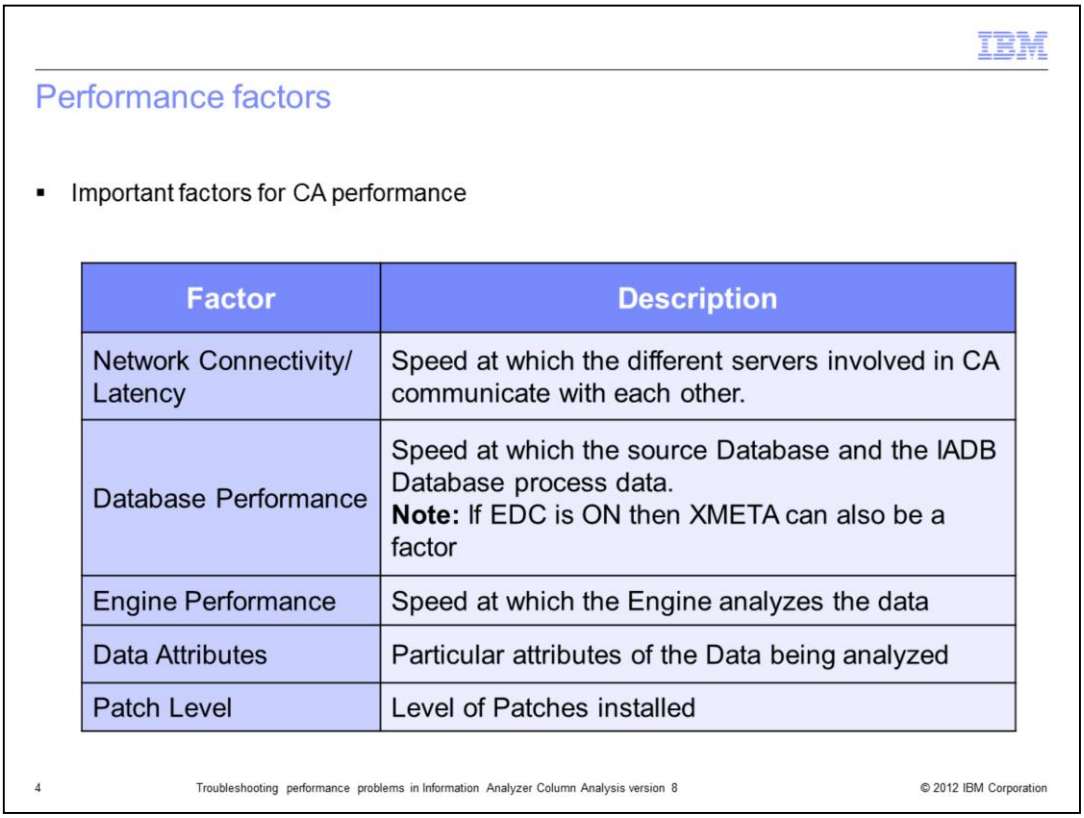


The objective of this presentation is to explain the most important factors that impact Column Analysis performance. This presentation explains how to collect runtime data to identify bottlenecks, how to troubleshoot and diagnose the root causes, and how to resolve and prevent these problems. In this presentation, Information Analyzer is referred to as IA and Column Analysis is referred to as CA.

CA consists of three main steps, which are started sequentially. The first step is **pre-process**, which starts CA and prepares the execution of the next steps. The next one is the **main-process**, which runs the bulk of the data extraction, data analysis and creation of results. The last step is the **post-process**, which runs additional processes, such as Enhanced Data Classification, or EDC, depending on the options selected for CA.

The time each step takes can be obtained from the file iasServer*.log which is located under WebSphere/AppServer/profiles/<IS_HOME_PROFILE>. This slide displays an example of how the times are shown in this file.

This presentation will focus only on analyzing performance problems that affect the main-process, which is the longest process and the most likely step to experience performance issues. If you observe performance problems involving the pre-processing step, engage IBM Support for assistance. If you observe problems with the post-processing step, check the section “Troubleshooting: XMETA with EDC on”.



The performance of the CA main-process can be affected by several factors. This slide displays the most common and important factors.

The first factor, **Network Connectivity/Latency,** refers to the speed at which the different servers involved in CA communicate with each other. Depending on your configuration, there can be up to four different servers working together: the engine, the XMETA repository, the IADB repository, and the Source Database Server.

The next factor, **Database Performance**, refers to the speed at which the Source Database and the IADB Database process data. It is important to note that if EDC is used, the performance of the XMETA database can also be a factor.

The next factor, **Engine Performance**, refers to the speed at which the engine analyzes data.

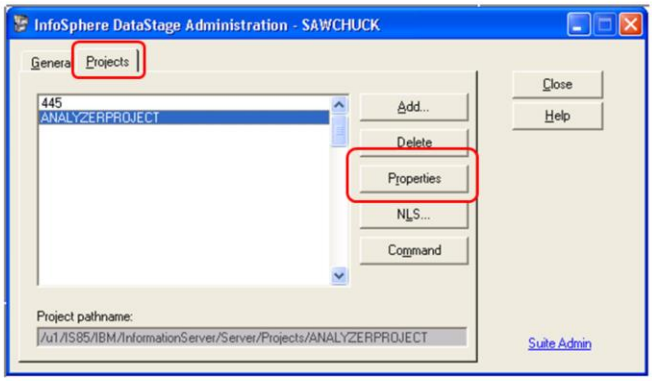
The next factor, **Data Attributes**, refers to the particular attributes of the data being analyzed. Different data types or large data volumes can take longer times to be analyzed.

The last factor is the **Patch Level.** If you are experiencing performance problems always make sure that you are running the latest Fix Pack and Rollup Patch level available for all your tiers.

This presentation will show how to diagnose and identify which of these factors can be impacting the performance of CA, with the exception of the Patch Level factor, as this varies over time and per version.

Collecting runtime data (1 of 13)

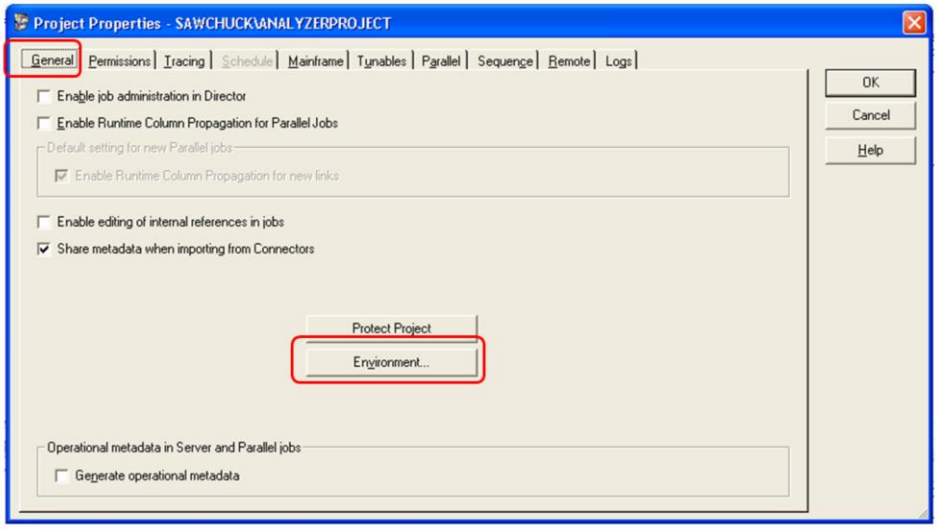
- Open DataStage® Administrator
- Login as DataStage administrator user
- Click Projects tab
  - Select ANALYZERPROJECT
- Click Properties



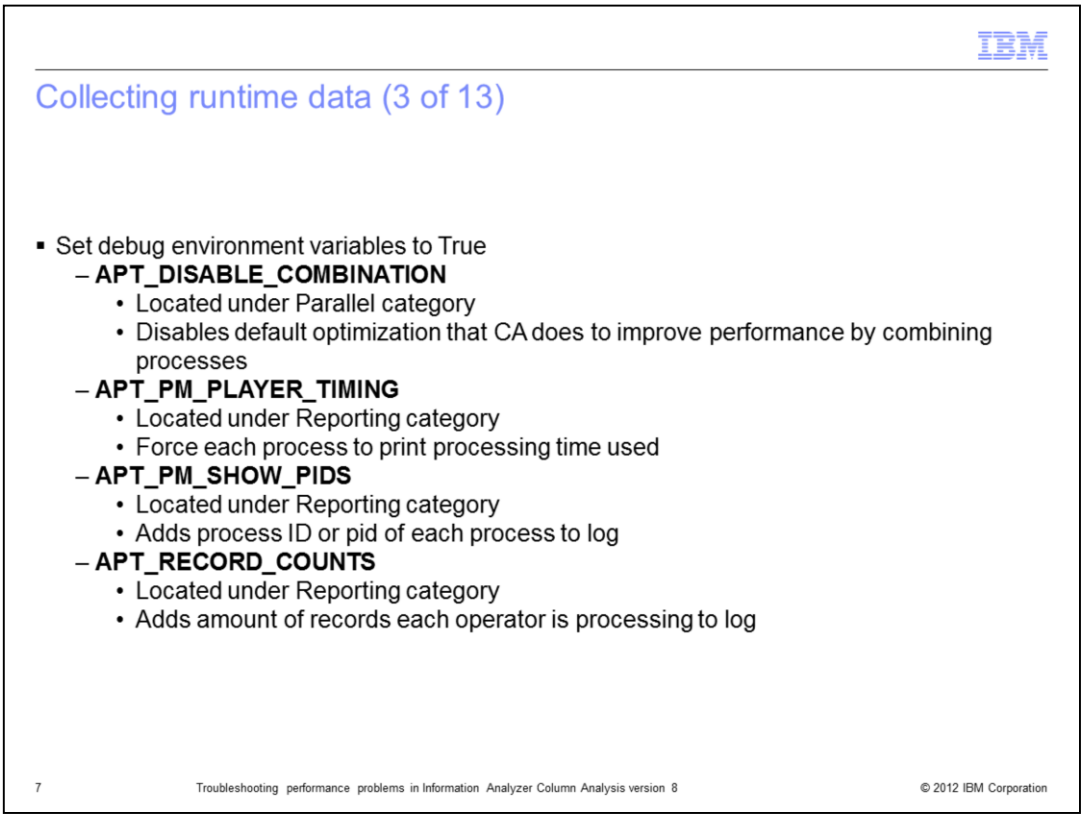
To troubleshoot performance issues, you first need to collect runtime data. This is done by setting debug environment variables at the project level for the ANALYZERPROJECT. To do this, you will need access to DataStage Administrator and DataStage Director. DataStage Director is not included with Information Analyzer and will require a separate license to use it. Open DataStage Administrator, login with a DataStage administrator user, click the Projects tab, select the ANALYZERPROJECT and click Properties.

Collecting runtime data (2 of 13)

- Go to General tab
- Click Environment



Go to the General tab and click the Environment Button to open the list of environment variables for this project.

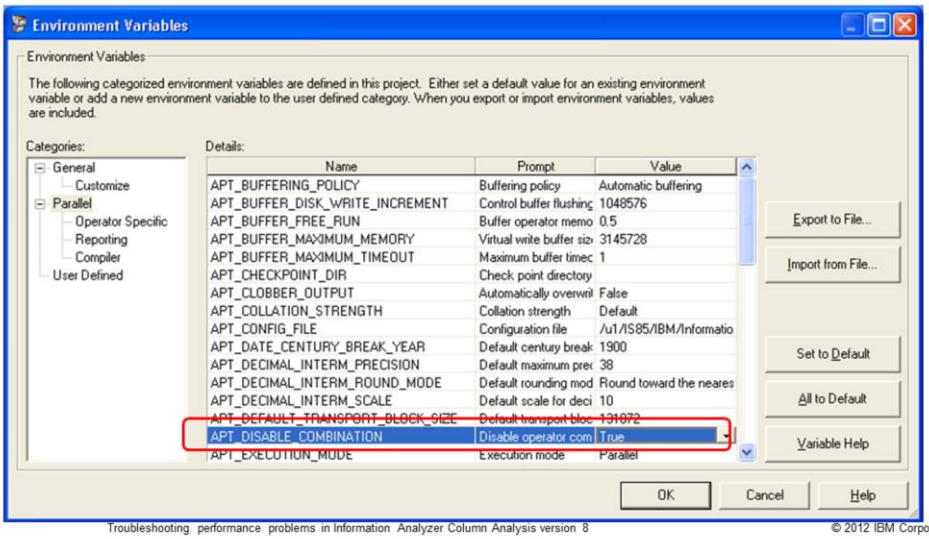


There are four environment variables that should be set to True to collect runtime data relevant for performance issues. The first variable is **APT_DISABLE_COMBINATION** which is located under the Parallel Category. Setting this to true will disable the default optimization that CA does to improve performance by combining processes. By disabling this, you may experience some performance decrease but the logs will display more detailed information. Next, set **APT_PM_PLAYER_TIMING** which is located under the Reporting Category. This variable will force each process to print the processing time used to the log file. Next, set **APT_PM_SHOW_PIDS** which is located under the Reporting Category. This variable will add the process ID, or pid, of each process. This can be useful if there is a particular process that is taking significantly longer than others so you can trace this process at the OS level. The last environment variable to set to true is **APT_RECORD_COUNTS** which is also located under the Reporting Category. This variable will add the amount of records each operator is processing to the log file. This is useful to ensure that the data is evenly distributed on each operator.

The next slide displays how to modify these environment variables.

Collecting runtime data (4 of 13)

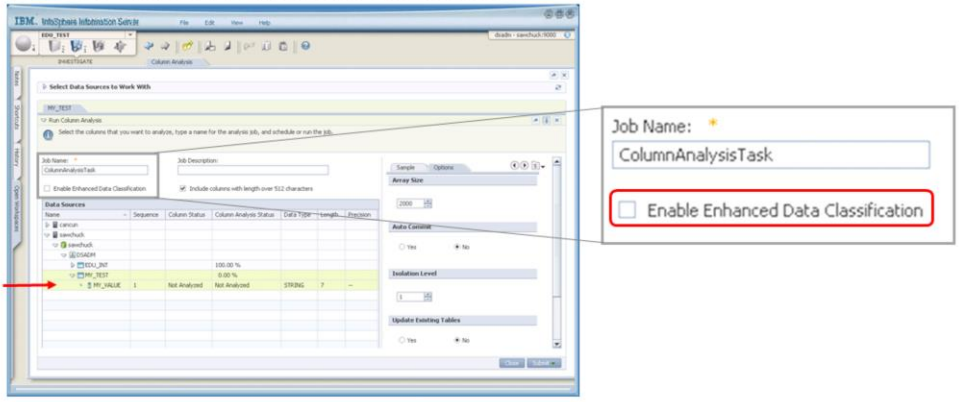
- Set each environment variable
- Save changes



To set these environment variables, find the variable you want to change then click the dropdown box to the right of the variable name and set the value to True. After having updated all the variables, click OK to save the changes. Click OK again to get back to the list of projects. It is not necessary to restart any component for the changes to take effect. All new CA submissions will use this configuration. Remember to set these variables back to False after you are done collecting runtime data.

Collecting runtime data (5 of 13)

- Select columns to analyze
- Uncheck “Enable Enhanced Data Classification”

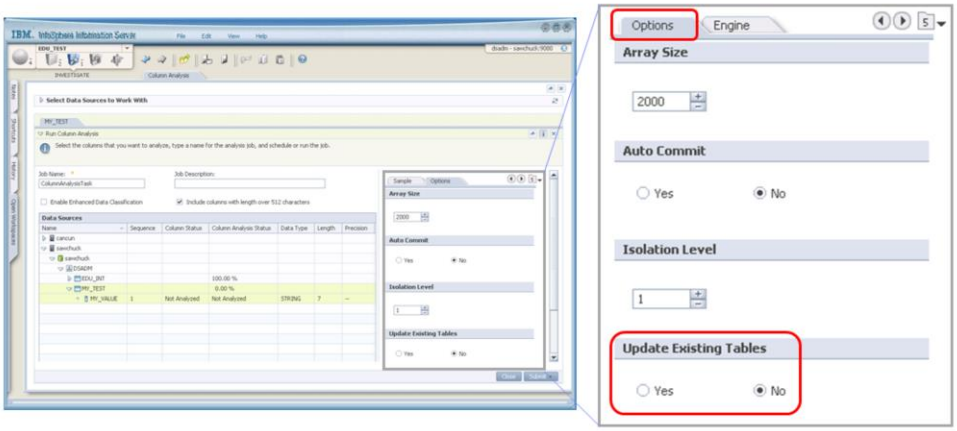


9 Troubleshooting performance problems in Information Analyzer Column Analysis version 8

Open the IA Client and select the column or columns you want to analyze. On the central panel, uncheck “Enable Enhanced Data Classification” if you see this option available. Older versions of IA did not have this option. Enabling this feature will add a process during the post-process step which will increase the time it takes for CA to complete. To isolate problems related only to the main process, disable this feature. If you have reasons to believe that this feature is causing performance problems then skip to the slide “Troubleshooting: XMETA when EDC on”.

Collecting runtime data (6 of 13)

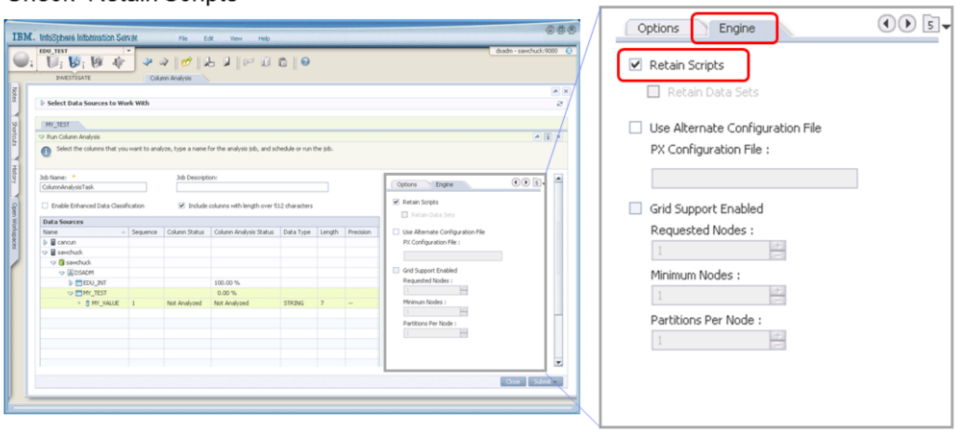
- Set “Update Existing Tables” to No



On the right panel go to Options and set “Update Existing Tables” to “No”. When troubleshooting performance issues, it is recommended not to update existing tables because database updates are slower than inserts. Setting this to No, forces CA to insert results into empty new tables which is a faster process. Set “Update Existing Tables” to “Yes” only if you want to keep previous results and you are not concerned about performance.

Collecting runtime data (7 of 13)

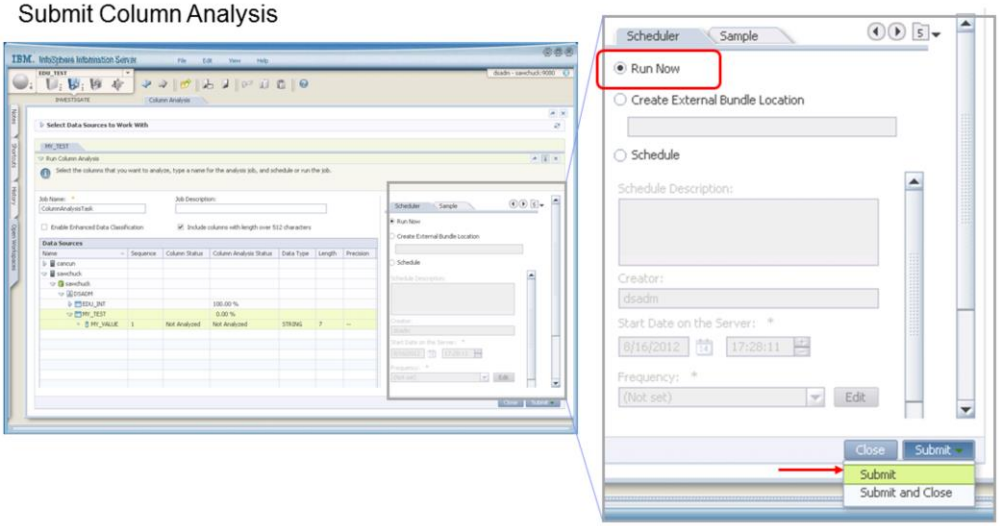
- Check “Retain Scripts”



Next, click the Engine tab and check the “Retain Scripts” box. This will keep additional files that can be used later to troubleshoot problems.

Collecting runtime data (8 of 13)

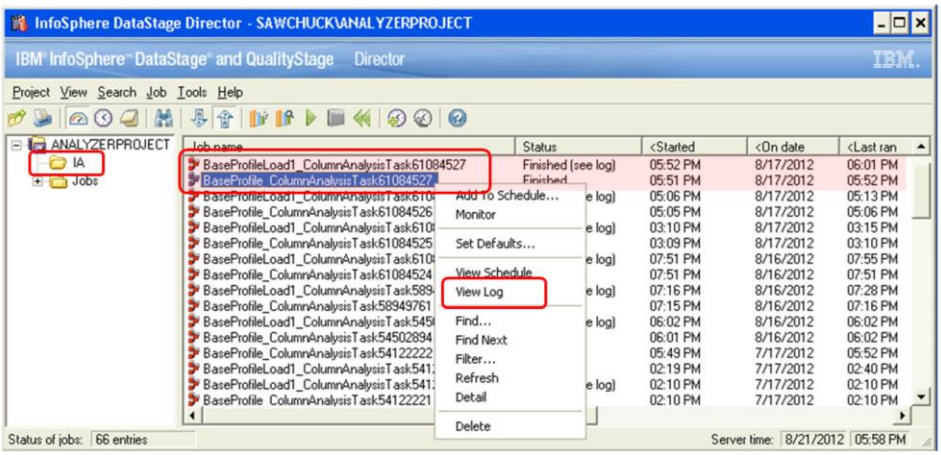
- Submit Column Analysis



Next, submit the Column Analysis.

Collecting runtime data (9 of 13)

- Open ANALYZERPROJECT project in Director
- Find Jobs created by Column Analysis
- Open logs

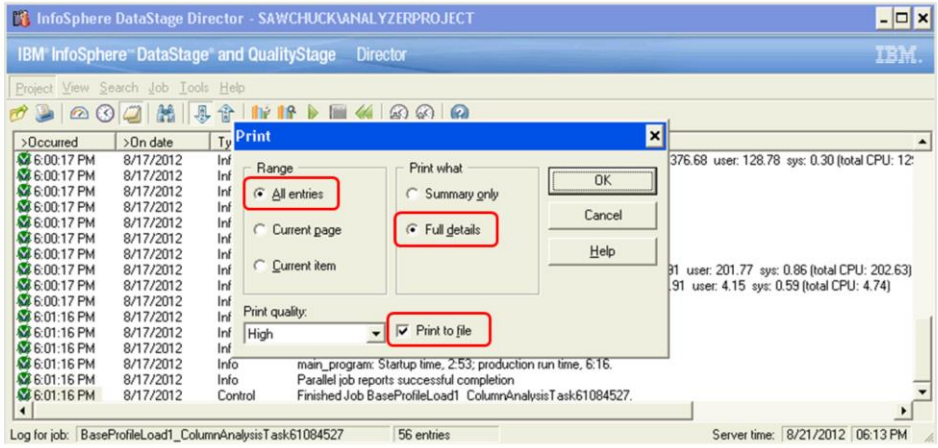


Next, open the DataStage Director and connect to the ANALYZERPROJECT. Select the IA folder and sort the Jobs by the “On Date” Column. Use the timestamps to identify the most recent jobs. CA creates two types of jobs, one with the prefix “BaseProfile” to extract the data from source database and a second type with the prefix “BaseProfileLoad” to analyze the data and insert the results into the IADB database. The number of actual jobs that CA creates, depends on the number of columns submitted for analysis. Ten columns or fewer will only create two jobs, one BaseProfile and one BaseProfileLoad job. More than ten columns will create additional jobs.

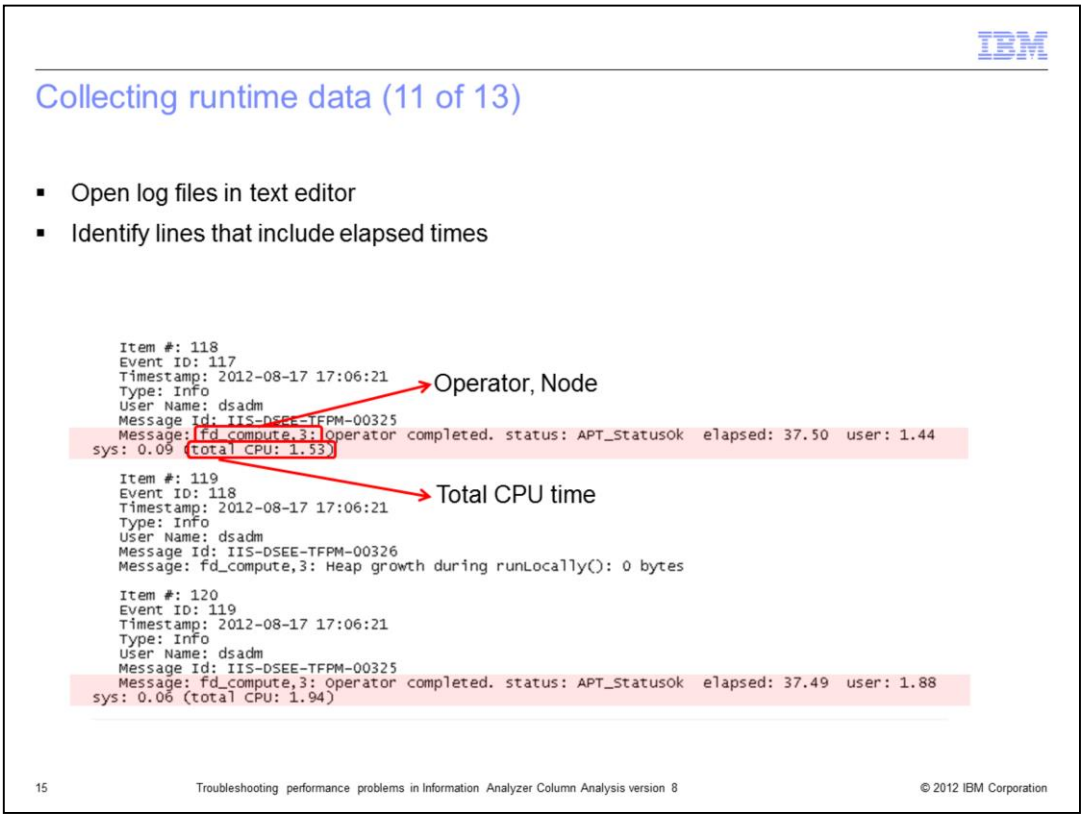
Right click the “BaseProfile” log and click “View Log” to open the log.

Collecting runtime data (10 of 13)

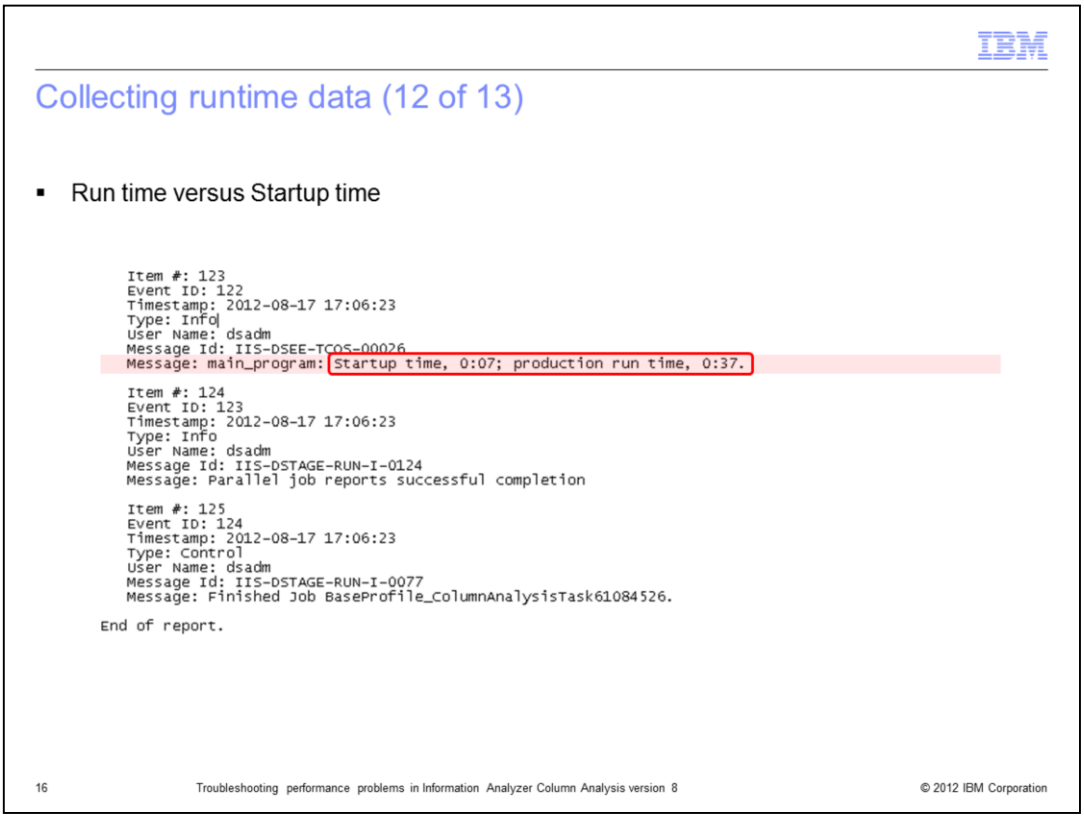
- Save log to a file



The next step is to save the log to a file. Click “Project” in the Menu bar, select the options “All entries”, “Full details”, and “Print to file”. Click OK and save the file. This will create a text file with the contents of this log. Repeat this process for the “BaseProfileLoad” log.



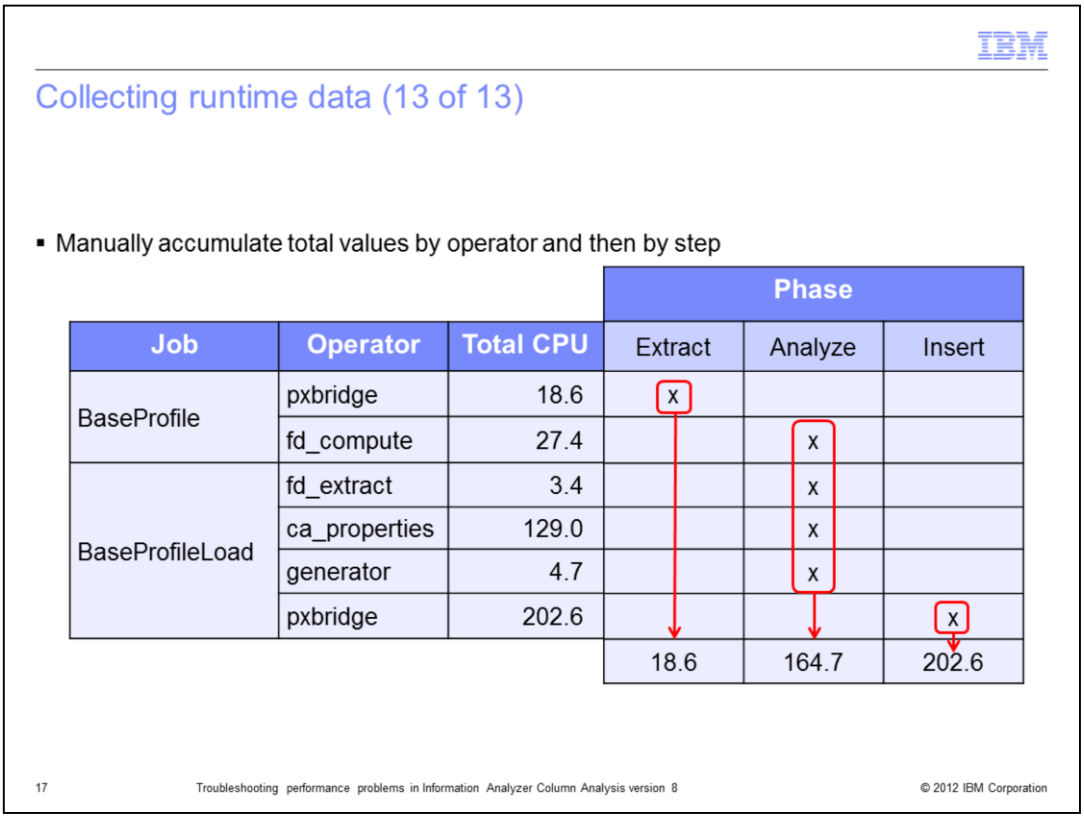
Next, identify the processing time associated with each operator. An operator represents a part of the job that is responsible for a specific task, for example, to extract data. Open the log files in a text editor and identify the lines that include the string “Operator completed”. These lines will include the operator name, the node and the “total CPU” time, expressed in seconds, for the operator. Write down all “total CPU” times shown in the log and add them together by operator. The order in which these values appear in the log is not relevant for this analysis.



Towards the end of the log you will see the **Startup and Production Run Time** values. The **Startup time** refers to the time it takes for DataStage to start all the processes of the job. The larger the number of nodes used in the configuration file, the greater the Startup time will be. The **production run time** is the elapsed time from the moment the first operator starts to run until the last operator finishes running. Production run time includes idle times. If you add these two times together you will get the amount of time the job took to complete. The processing times that each operator reports and that was discussed in the previous slide occur during the production run time part of the job. However, keep in mind that processing times are reported individually by each operator and operators can run in parallel. Because of this, the total processing time of all operators is typically larger than the production run time shown by the log. However, the opposite is also possible because “Total CPU” times do not include idle times between processes and in busy systems the Production Run time can be larger.

As a result of this, Total CPU times by operators should only be used as a reference to understand how the processing time is being distributed by step and not as a way of calculating the actual time it took the job to complete. To calculate that, add the Startup time plus the production run time.

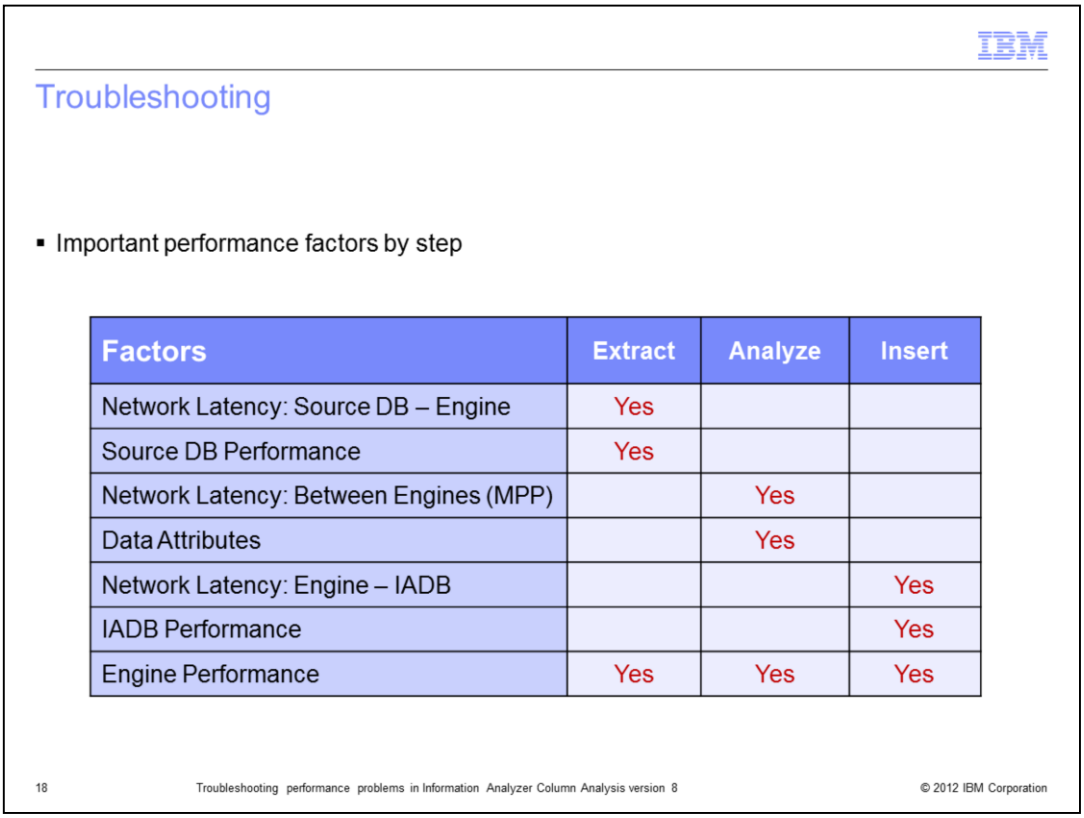
The next slide displays how to use the total processing times by operator to identify bottlenecks.



For the purpose of this presentation, the production run times of the operators is classified in three phases. The first phase, **Extract**, includes operators that connect to the source database and extract the data to be analyzed. The next phase, **Analyze**, includes operators that look at the extracted data and analyze it to find data formats, data types, patterns, null ability, and so on. Finally, the **Insert** phase includes operators that take the results generated during the Analyze step and inserts them into the IADB repository.

Using the total processing times by operator, you can now calculate a metric to estimate how much processing time took each phase. The table displayed on this slide shows how to assign each operator's processing time to a specific phase. For example, the "pxbridge" operator in the "BaseProfile" job belongs to the Extract phase.

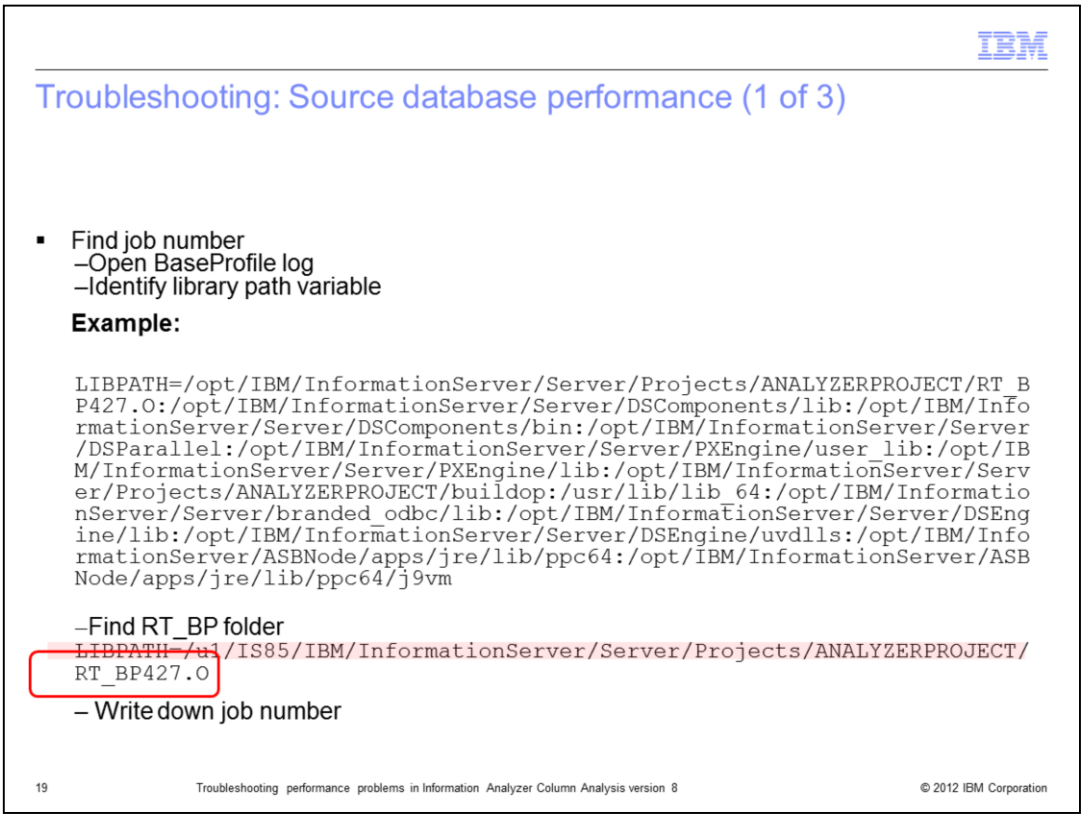
Use this table to compare different executions of CA and understand how processing times per Phase behave after you change certain variables such as the number of nodes, the source database, or the data. You should repeat the steps in this exercise a few times changing one variable at a time to better understand what are the common bottlenecks.



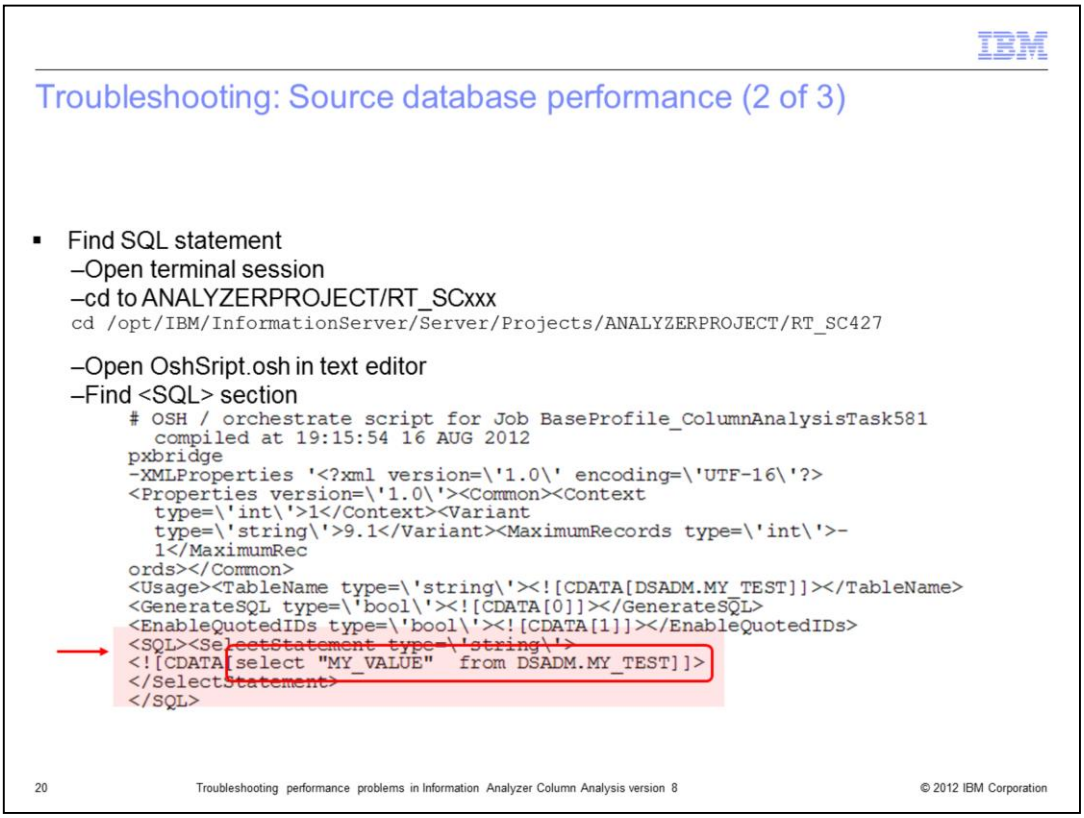
Once the bottlenecks are identified, the next step is to troubleshoot the possible causes. The table displayed on this slide shows the main performance factors for each phase. For example, if the “Insert” phase is taking too long then check three things: the “Network Latency” between the Engine and IADB, the IADB Performance, and the Engine Performance.

Be aware that a large value for a phase, in particular for the Analyze or Insert phases, does not necessarily mean that you are experiencing a performance problem. Analyzing very large tables with high cardinality can take a lot of time to complete. Before drawing any conclusions about performance, run and compare multiple executions to make sure that you are observing a consistent behavior.

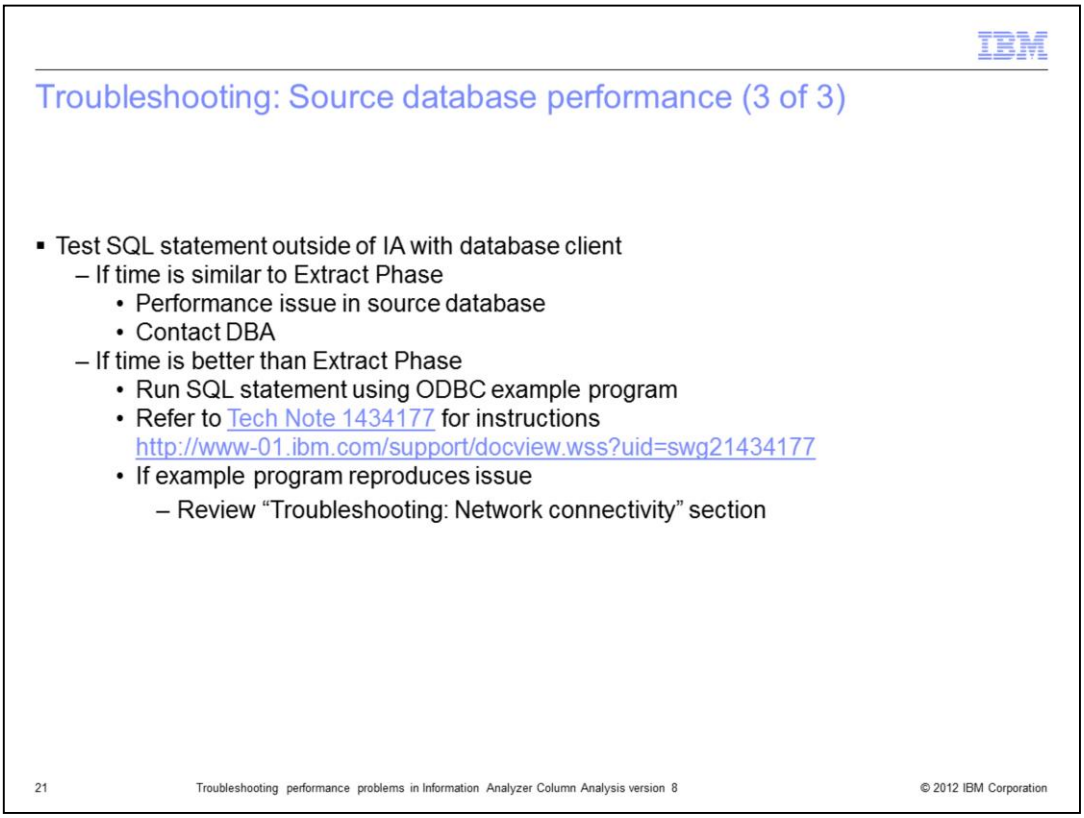
The next slides display how to troubleshoot each factor.



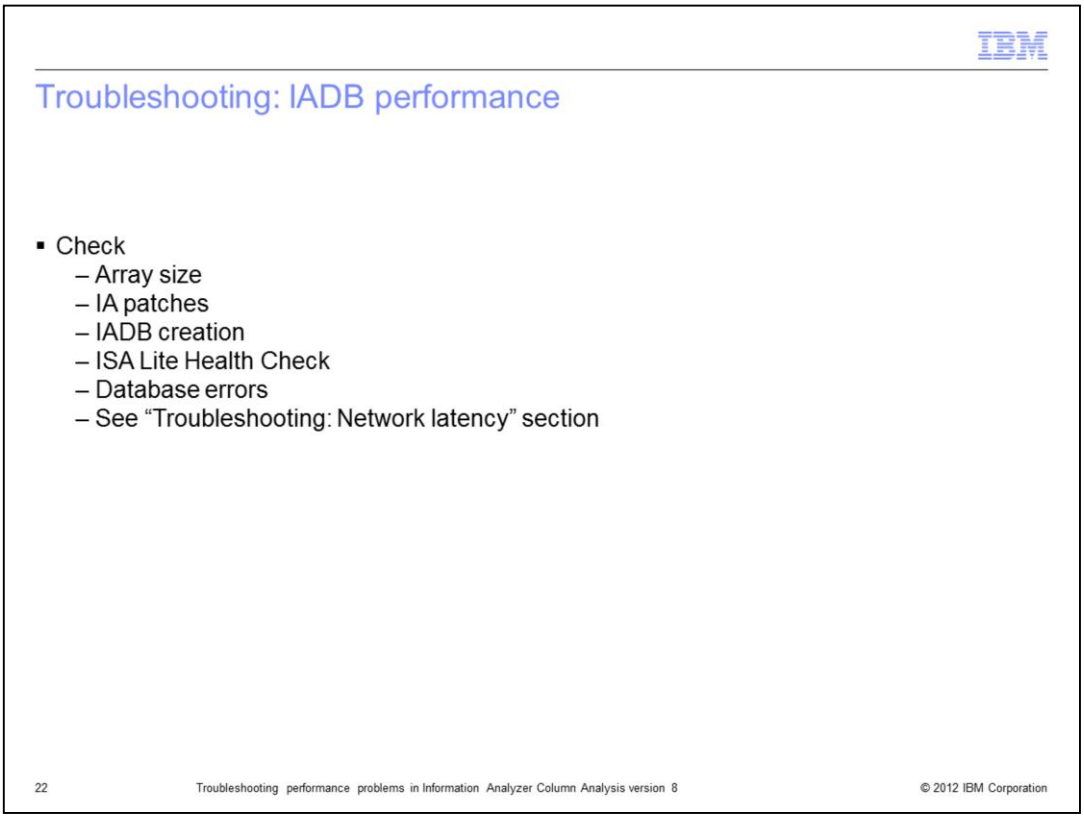
To confirm if there is a performance problem on the source database you need to test the SQL query that IA is running to extract data. To find this SQL query, you first need to identify the job number. The easiest way to do this is to open the BaseProfile log in a text editor and identify the line that includes the library path variable. Environment variable values are listed in the second entry of each log. This variable can be named LIBPATH, LD_LIBARY_PATH or SHLIB_PATH depending on your operating system. Inspect the value of this variable and identify the path that includes the RT_BP folder. This typically appears at the beginning of the entry. Write down the number after the prefix RT_BP. This is the job number. In this example, the number is 427.



Next, open a terminal session and change directories to the RT_SC folder that ends with the job number you obtained in the previous slide. In this example, it is RT_SC427. Open the OshScript.osh file in a text editor and find the SQL tag. This will contain the query that CA is using to extract data.



Run the SQL statement outside of IA using a database client application and compare the times it takes to complete the query. If you obtain a time similar to what you got in the Extract phase, then the performance problem is coming from the source database. Contact your DBA to discuss the performance of this source database. If you obtain a significantly better time using a database client application, then run the SQL using the "example" program from the Engine Server. Refer to Tech Note 1434177 for instructions on how to do this. This will use the same type of connectivity that IA is using to extract data. If the SQL statement executed with the example program reproduces the slow performance you get in IA, review the section "Troubleshooting: Network connectivity".



Next, this presentation discusses troubleshooting IADB performance. IADB performance can be impacted by several factors. First, try to increase the array size located in the Options tab of CA. Higher values for this parameter can improve the performance of inserting rows to IADB. Try doubling the value and compare results. If you observe an improvement, you can continue doubling value until you find an optimal setting. If the array size is set too high, performance will begin to decrease.

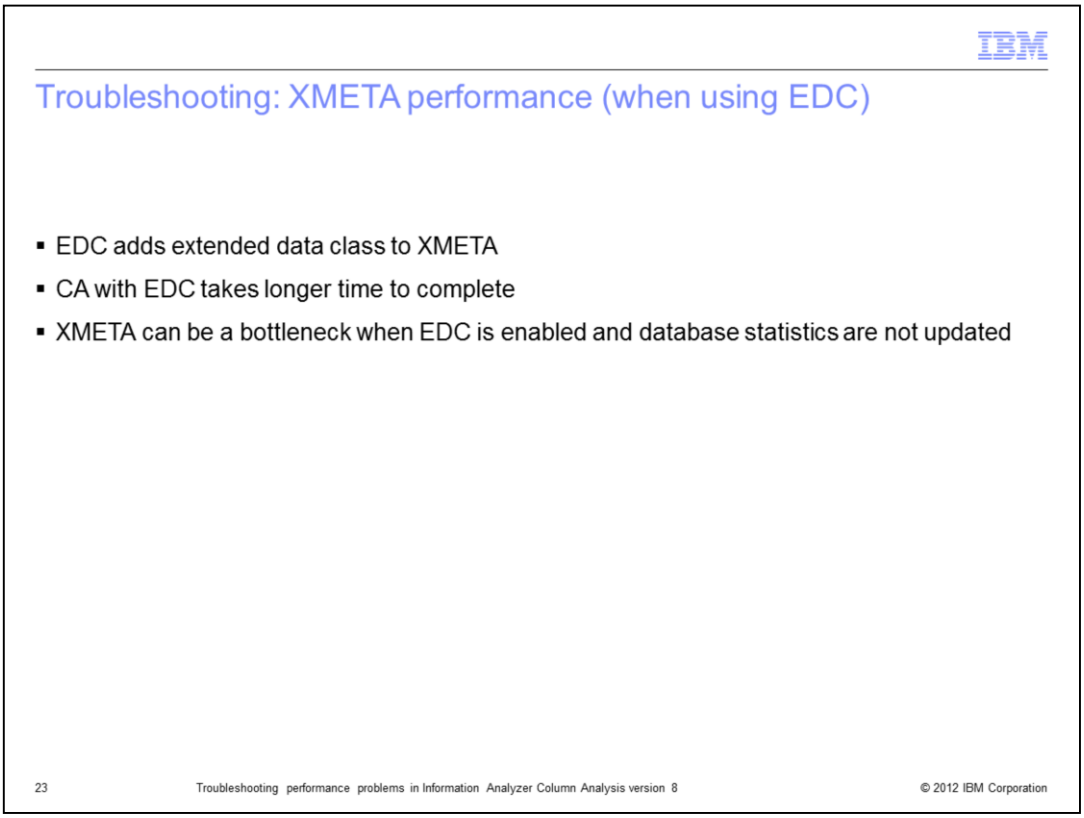
Next, make sure that you are running the latest patches for the version of IA that you are using. New patches may introduce improvements in performance.

Next, confirm that the IADB database was created using the scripts provided with the installation media. If IADB was created without following the documented procedure, then IADB needs to be re-created.

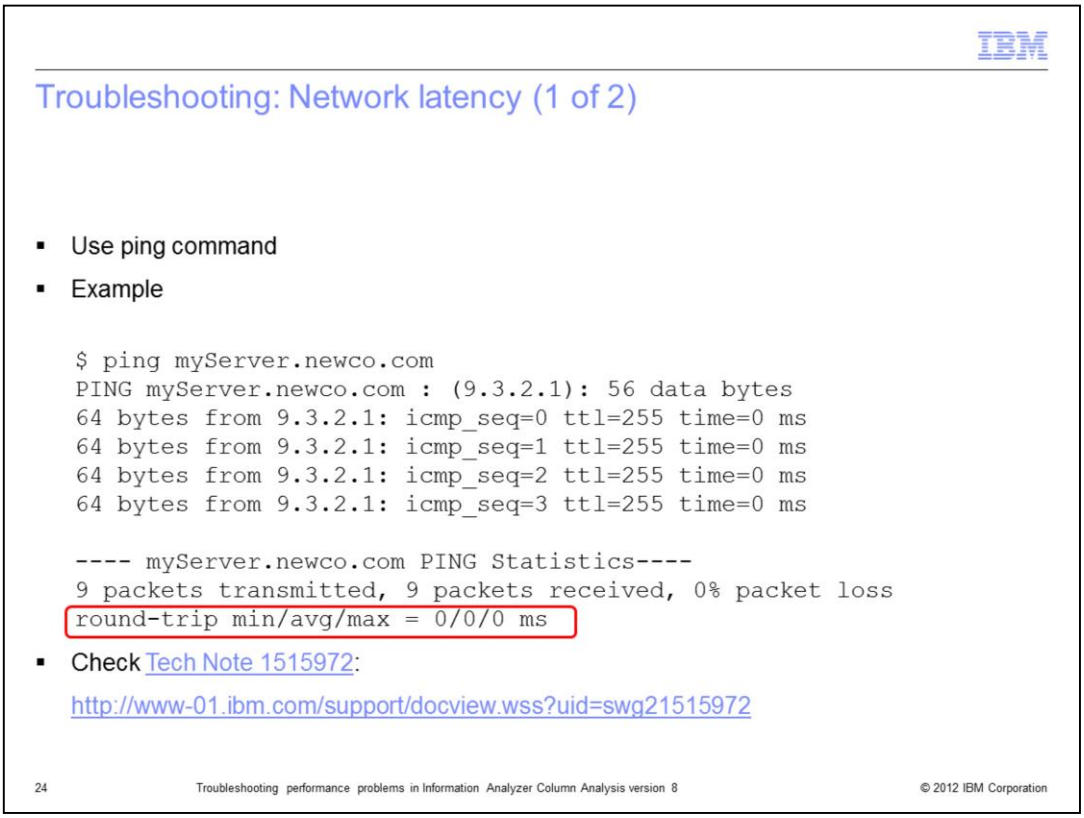
Next, check that there are no errors for IA shown in the ISA Lite Health Check. For more details about ISA Lite, see Tech Note 4022700.

Next, check database logs to confirm that there are no errors, such as lack of space. Check with your DBA to obtain this information.

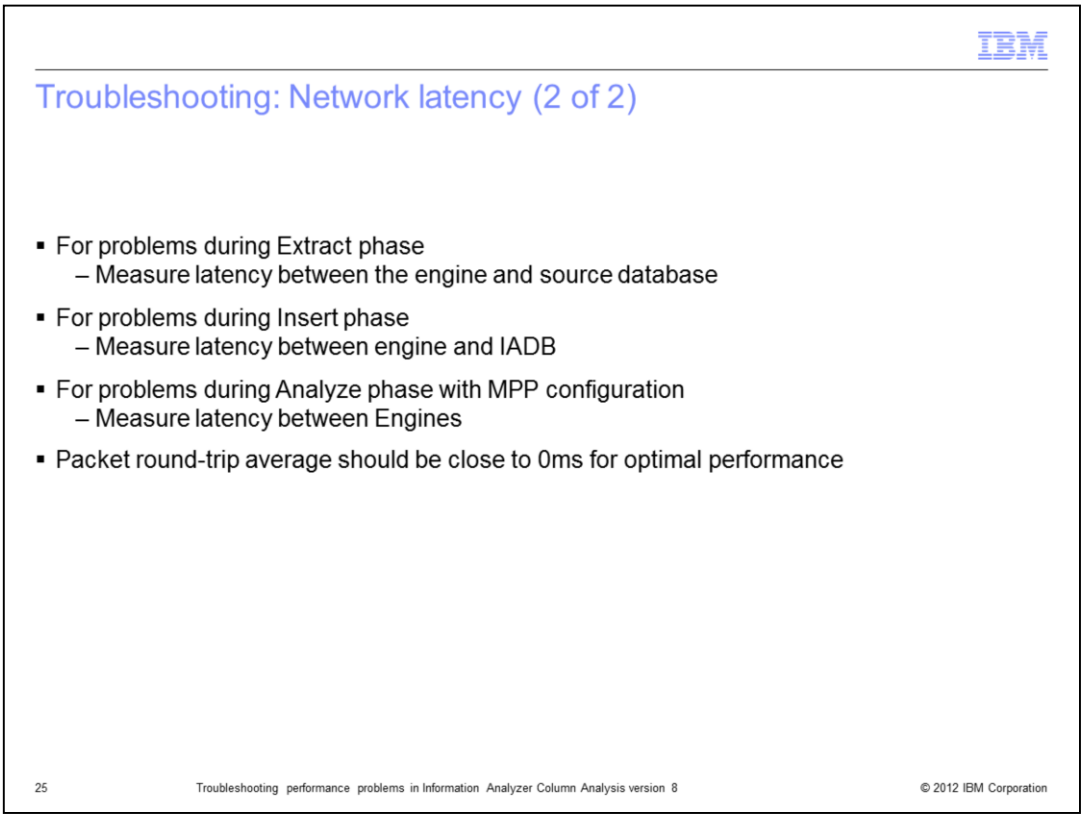
Finally, if the IADB is on a different machine than the engine, check the section “Troubleshooting: Network latency” to measure the latency between the engine and the IADB.



Another performance issue may be caused by EDC. EDC is a feature that identifies the extended data class of each distinct value of a column. When EDC is enabled, CA will take longer time to complete; this is the expected behavior. However, some customers have reported significant performance problems after adding EDC. These problems were caused by XMETA not having updated database statistics. If you are having performance problems that occur only when EDC is enabled, contact your DBA to make sure that RUNSTATS has been executed recently to update the statistics of the XMETA database.



Network connectivity can also impact the performance of CA. To investigate if this is a problem, measure the speed of a data packet round-trip using the ping command. This slide displays an example using the ping. For a more accurate way to measure network latency, see Tech Note 1515972

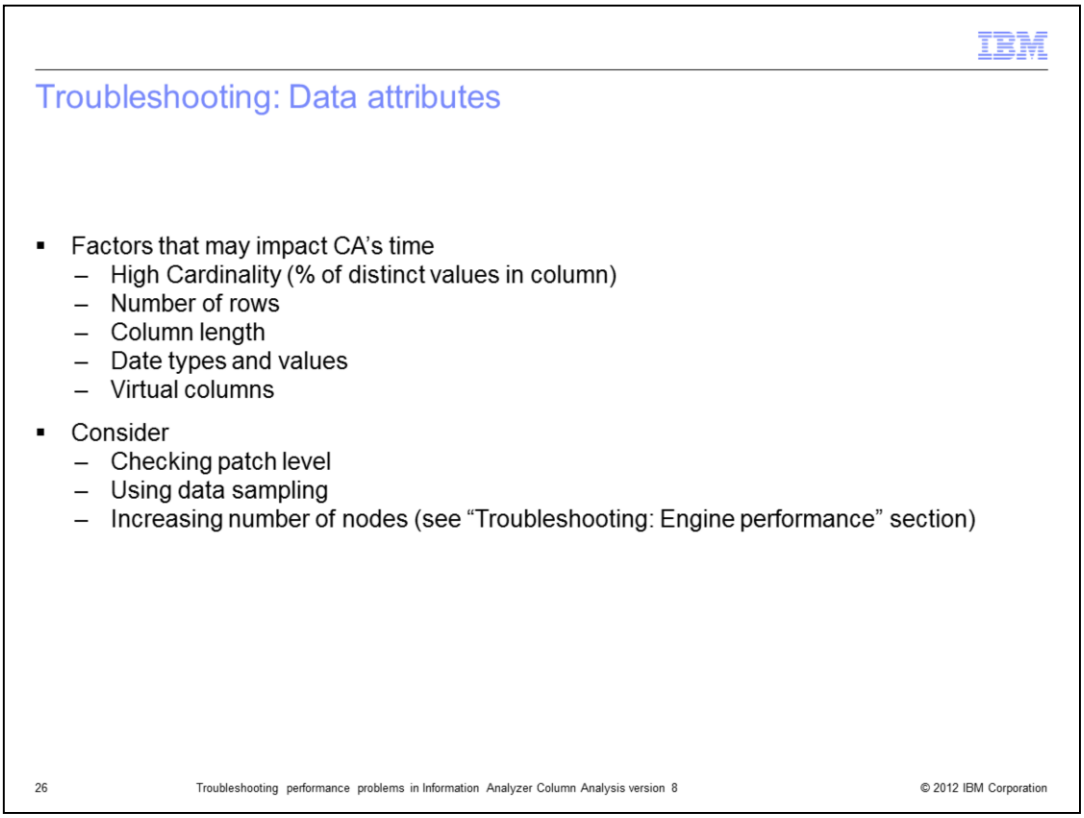


If you are experiencing problems during the Extract phase, measure latency between the engine and the Source Database.

If you are experiencing problems during the Insert phase, measure latency between the engine and the IADB.

If you are running an MPP configuration and you are experiencing problems during the Analyze phase, measure latency between engines.

Ideally, a packet round-trip average should be close to 0ms for optimal performance. Larger values will impact the communication between the servers. If you measure high values for this metric, contact your Network administrator to discuss ways to improve this.



The characteristics of the data you analyze also have a significant impact in the overall performance of CA. This is the expected behavior of the product. There are several data attributes that can impact CA's performance.

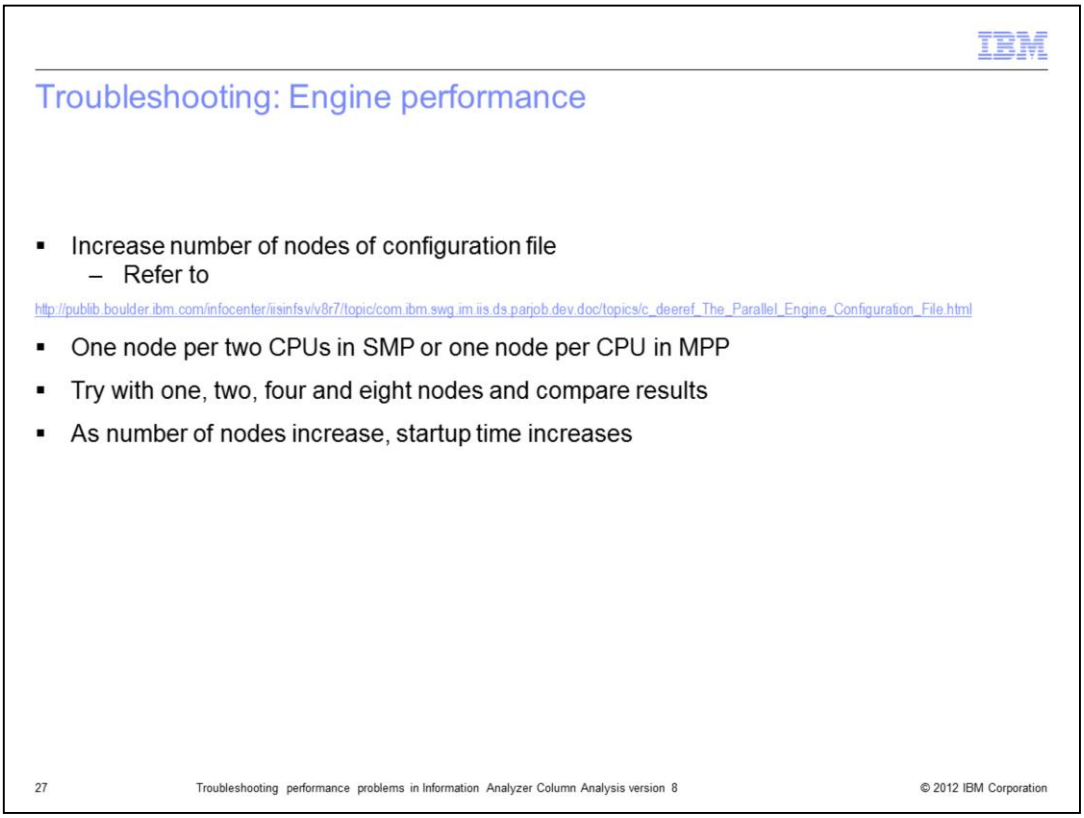
High cardinality, which is defined as the percent of distinct values in a column, may have an impact. The higher the cardinality of a column, the longer it will take to be analyzed. Number of rows may also have an impact on performance. The larger the number of rows of a table, the longer the data will take to be analyzed.

Column length is another contributing factor. The more bytes a column has the longer it will take to be analyzed. This impact is only noticeable when comparing columns of very different lengths.

Data types may also be a factor as certain values of data types, such as dates or strings that can be interpreted as dates, will take longer time to be analyzed.

Another factor that may cause performance problems are virtual columns. These are columns that do not exist in the source database and are defined in IA. These columns will require extract calculations and will increase the time to complete CA.

If you identify that one of these factors is associated with significantly slower times, make sure that you are using the latest patches available. Try to use data sampling to reduce the number of rows you are analyzing and increase the number nodes as explained in the next slide, Troubleshooting: Engine performance.



You can improve the performance of the engine by increasing the number of nodes in the configuration file used by IA. For instructions on how to do this, see the online documentation at the link displayed on this slide. The rule of thumb is to have one node per each two CPUs in an SMP configuration. In an MPP configuration, you can use one node per CPU. This is a suggested starting point. Modify the number of nodes and compare values to find what works best in your environment.

Run tests using one, two, four and eight nodes and compare the results.

Keep in mind that the more nodes you have, the greater the startup time will be. This time, it is not included in the processing times reported by operators, which refer to run time, so when tuning the number of nodes, take into account this time as well.

Trademarks, disclaimer, and copyright information