Understand Informix High CPU Utilization

Contents

Introduction

This document describes how Unified Contact Center Express (UCCX) activities, which require local UCCX database access, might perform slowly.

Background Information

It causes AppAdmin pages to load slowly, updates to AppAdmin to take a long time to take effect, a delay in the response to a wallboard query, Workforce Manager to be unable to query UCCX data, and other performance and stability issues.

The command `show process load` , entered in the CLI , shows that the `uccxoninit` consumes a large amount of CPU. The `uccxoninit` process represents the UCCX Informix database instance which runs on the UCCX server.

Feature Information

The database engine that supports the UCCX application is Informix from IBM. Configuration and historical information that is added to UCCX 's AppAdmin page and is produced by the UCCX application is stored in the UCCX Informix instance.

The UCCX application provides three users that can be used to access the UCCX database directly in order to extract information for the purposes of wallboard applications, Quality Management, Workforce Management, and custom historical reporting.

User information, permissions for each user, and the intended purpose of each user are described here:

- `uccxhruser` - This user has select permissions to many configuration and historical tables in the UCCX database and can be used only for custom historical reporting and Cisco Unified Workforce Management (WFM). Queries and stored procedures executed by this user can perform complex, long-running queries. Due to the profile of a typical historical reporting or WFM user, these queries and stored procedures cannot be executed frequently as would occur for a wallboard application.

  Although many wallboard applications require data contained within the configuration and historical tables to which the `uccxhruser` has access, it is technically not supported to use this user to execute complex, frequent queries against the UCCX database for the purposes of a wallboard application.
- `uccxworkforce` - The `uccxworkforce` user has access to the Team, Resource, and Supervisor tables which must be used for Cisco Unified Quality Management (QM). Workforce Management must use `uccxhruser` as it requires access to historical data tables that are not accessible by the `uccxworkforce` user.

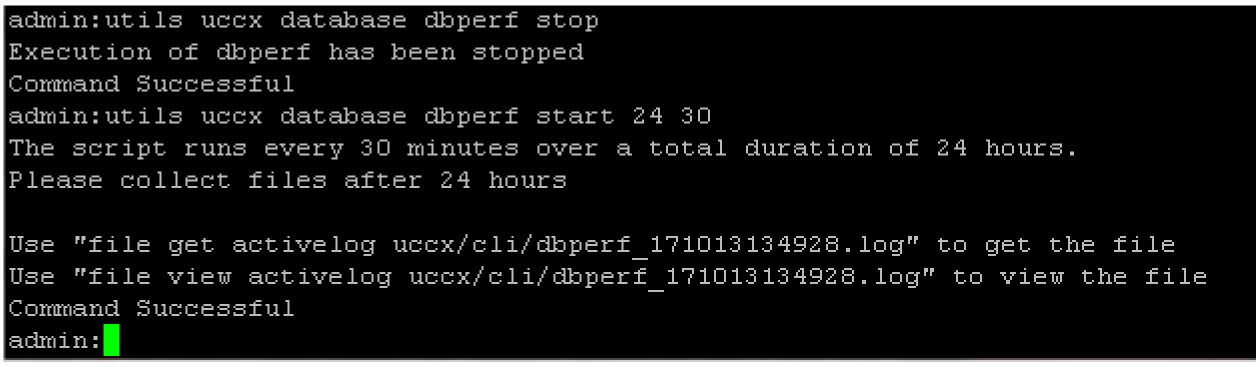
- uccxwallboard - This user has select permissions only on the real-time database tables that contain snapshots of real-time statistics written from the memory of UCCX Engine. The select permissions restricted to tables RTCSQsSummary and RTICDStatistics mean the uccxwallboard user can be used to query the UCCX database frequently with simple, non-complex queries intended to be sourced by a wallboard application.

Troubleshoot Methodology

In UCCX Release 10.0 and later, enter the utils uccx database dbperf start <totalHours> <interval> command in order to begin performance tracing on the UCCX database. Theinterval argument in this command determines the periodicity of the trace collection and the totalHours argument determines the total amount of time the tracing runs before it is disabled. These parameters are optional. If they are not specified when the command is executed the default values of 20 minutes and 10 hours are used.

For example, enter the utils uccx database dbperf start 24 30 command in order to enable performance tracing on the database and collect data on performance statistics every 30 minutes for 24 hours.

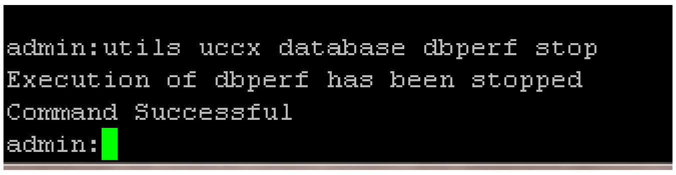
Instructions to collect the data obtained by the CLI command are printed in the command output.



```
admin:utils uccx database dbperf stop
Execution of dbperf has been stopped
Command Successful
admin:utils uccx database dbperf start 24 30
The script runs every 30 minutes over a total duration of 24 hours.
Please collect files after 24 hours

Use "file get activelog uccx/cli/dbperf_171013134928.log" to get the file
Use "file view activelog uccx/cli/dbperf_171013134928.log" to view the file
Command Successful
admin:
```

After the totalHours given, the data collection automatically stops. In order to manually stop the data collection, enter the utils uccx database dbperf stop command.



```
admin:utils uccx database dbperf stop
Execution of dbperf has been stopped
Command Successful
admin:
```

If the UCCX version is Release 9.0(2) or earlier and the utils uccx database dbperf command is not available, contact the Technical Assistance Center (TAC) for further assistance.

TAC executes the dbperf.sh script attached to Cisco bug ID CSCuc68413 manually with Remote Support Account access.

When you determine when to start the script execution either manually or through the CLI command, the periodicity, and the total time, ensure the CPU consumed by the uccxoninit process fluctuates significantly or remains high during those periods in order to collect the necessary information for root cause analysis.

Additionally, periodically enter the show process load command to determine when the CPU fluctuates in order to correlate the logs collected by the dbperf tracing script.

Data Analysis

The logs collected by the dbperf script execution of onstat -g ses 0 show active queries that are issued against the UCCX database. High CPU on the uccxoninit process is typically the result of complex queries that take a long time to execute. The goal is to determine the queries that consume the most resources, determine the source client for those queries, disable the queries from the client for immediate resolution, and optimize the long-running queries for permanent resolution.

In the logs collected by the dbperf script, look for queries that most likely cause high fluctuations in CPU or sustained high CPU consumption by the uccxoninit process.

Suspect queries:

- Are issued from sessions connected as uccxhruser - As described earlier, uccxhruser has privileges to select information out of a vast number of configuration and historical tables. As a result, complex, long-running queries across multiple tables can be constructed and can have performance impacts on the UCCX database. Although not absolute, uccxwallboard and uccxworkforce users have such limited access to tables within the UCCX database, complex queries that cause performance impact issued by these users are unlikely. Additionally, queries issued byuccxhrc are issued by the UCCX Historical Reporting Client (HRC) or Cisco Unified Intelligence Center (CUIC) against the UCCX database. These queries are static and cannot be modified and the queries, along with relevant indices, have been written, tested, and tuned for minimal performance impact.
- Perform intensive queries on historical tables - Queries that require the UCCX database to perform multiple joins across tables, select significant amounts of information or operate on non-indexed fields could cause performance impacts to the UCCX database.

An example with a complex query that involves an HR table run as uccxhruser is shown here:

```
 session                                     # RSAM     total      used       dynamic
 id       user      tty        pid        hostname  threads  memory     memory     explain
 435050    uccxhrus  WBBOX 836      10.16.5. 1        90112      80712      off

..................

 Current  SQL  statement :
   SELECT x. resourceName , t. eventType , x. datetime , x.extension FROM ( SELECT
     t1. resourceID , t1. resourceName , t1.extension, MAX(t2. eventDateTime ) AS
      datetime  FROM  Resource AS  t1,  AgentStateDetail  AS  t2 WHERE  t2. agentID
     = t1. resourceID  AND t1. assignedTeamID   = 21 and t1.active GROUP BY
     t1. resourceID , t1. resourceName , t1.extension ) AS  x,  AgentStateDetail  AS
     t  WHERE  t. agentID   = x. resourceID  AND t. eventDateTime   = x. datetime
     ORDER BY  x. resourceName
```

This example shows a complex query, entered by uccxhruser sourced from the host WBBOX that could cause a performance impact on the UCCX database if it was entered often or was entered periodically before the previous query had returned results.

Although rare, UCCX database performance can also degrade (and the CPU utilization of the uccxoninit process fluctuates or remains high), as a result of the built-in purge process. The purge process is designed to delete data from the configuration and historical tables within the UCCX database in order to maintain the size of the database. Purge can be scheduled based on the size of the database or the oldest record contained within the database.

When the purge process runs, the data is removed with one query. It is not done iteratively based on the number of records to remove. This means that if the purge detects a large amount of data that must be removed, it issues a single query in an attempt to remove this data.

The modification of the purge schedule or parameters from the UCCX AppAdmin page in order to schedule the purge to remove a large amount of data can cause this single query, upon the next scheduled purge, to take a significant amount of time to complete. Therefore, it drives up the CPU utilization of the database instance.

In the output of the dbperf script, the purge query can be seen. It must be the only query entered by the user uccxuser that calls the sp_purge stored procedure.

```
session                                     # RSAM     total      used       dynamic
id       user      tty        pid        hostname  threads  memory     memory     explain
5628      uccxuser  -        -1       CC- EXPR - 1        544768     523408     off



Current  SQL  statement in procedure db_ cra : sp _purge
    proc -counter 0x0x4ccf9260  opcode   SQL


delete from  contactroutingdetail
 where (exists
   (select 1
      from  contactcalldetail  as  ccdr
      where (and (and (and (and (and (=  contactroutingdetail . sessionid ,
 ccdr . sessionid ), (=  contactroutingdetail . nodeid ,  ccdr . nodeid )),
(=  contactroutingdetail . sessionseqnum ,  ccdr . sessionseqnum )),
(=  contactroutingdetail . profileid ,  ccdr . profileid )), (>=  ccdr . enddatetime ,
p_ purgefrom )), (<  ccdr . enddatetime , p_ purgeto ))));
```

Common Problems

Based on recent Cisco TAC and Cisco Development Engineering experience, these are the most commonly seen issues which cause high CPU utilization on the uccxoninit process:

- A client in the enterprise connects as uccxhruser and runs frequent complex queries on the wallboard tables (RTICDStatistics and RTCSQsSummary) joined with the historical tables in order to provide a wallboard or custom reporting solution. For wallboard use, only use the uccxwallboard user and limit queries to the real-time tables. The ability to query the historical or configuration tables from a wallboard or with a frequency similar to a wallboard is not supported.
- A client attempts to execute custom historical reports on the active primary node instead of the secondary node. Only execute stored procedures, either custom or default, that produce historical reports on the standby node. CUIC and HRC execute queries on the standby node by default, but when it develops a custom historical report, the developer has a choice on which node to run these queries or execute these stored procedures.
- Reporting performance is found to be significantly degraded after a weekend. UCCX performs database maintenance activities at 3 AM server time every Saturday and Sunday. If the database tables are locked by continued queries at that time then it can cause the maintenance to fail and result in degraded performance until the next automatic maintenance. Avoid this by not running scheduled reports or queries against the UCCX database at 3 AM during the weekend. If reporting performance is impacted, contact Cisco TAC to manually perform this maintenance.

- Cisco Workforce Management (WFM) issues a complex query on the ContactRoutingDetail table in order to attempt to filter on the startdatetime field. No index is created on this field in this table by default, so the performance of this query is poor. WFM issues this query periodically in an attempt to synchronize data from UCCX to WFM. This issue is captured in Cisco bug ID CSCtz23710 and is resolved in WFM Release 9.0(1)SR4. Customers who experience this issue must upgrade to a version of WFM that contains a fix for Cisco bug ID CSCtz23710.

- Purge thresholds are modified such that the next scheduled purge attempts to remove a large amount of data. Rather than significantly modify the purge parameters in a single update, the purge schedule modifications are made iteratively, with a few days between purge configuration modifications. This allows the purge process to remove smaller sets of data in each pass, which improves the performance of the delete operation.
- The DialingList table is extremely large. The DialingList table stores all contacts uploaded to Outbound Campaigns. In UCCX Releases 8.0 and 8.5, after millions of records are uploaded to Outbound Campaigns, performance issues result then the table is queried (which causes high CPU on the uccxoninit process and makes the AppAdmin page slow). In order to mitigate the performance issues, open a TAC case for the installation of a cron job script that cleans up the DialingList table. In UCCX Release 9.0, an index was added to this table for more effective queries from AppAdmin in an attempt to improve performance. This change resolved the issue in all but the most extreme cases. In UCCX Release 10.0 the DialingList has been split into two tables, one for active contacts and another for historical contacts, which provides a comprehensive fix for this issue.