Uso elevado de la CPU de Informix

Contenido

Introducción

Este documento describe cómo pueden funcionar lentamente las actividades de Unified Contact Center Express (UCCX), que requieren el acceso a la base de datos UCCX local. Hace que las páginas AppAdmin se carguen lentamente, que las actualizaciones a AppAdmin tarden mucho tiempo en afectar, un retraso en la respuesta a una consulta de panel, que Workforce Manager no pueda consultar los datos UCCX, y otros problemas de rendimiento y estabilidad.

El comando **show process load**, ingresado en la CLI, muestra que la **uccxoninit** consume una gran cantidad de CPU. El proceso **uccxoninit** representa la instancia de base de datos de UCCX Informix que se ejecuta en el servidor de UCCX.

Contribuido por Sridhar Chandrasekharan, Ryan LaFountain y Ben Wollak, Ingenieros del TAC de Cisco.

Información sobre la Función

El motor de base de datos que soporta la aplicación UCCX es Informix de IBM. La configuración y la información histórica que se agrega a la página AppAdmin de UCCX y que se produce mediante la aplicación UCCX se almacena en la instancia de UCCX Informix.

La aplicación UCCX proporciona tres usuarios que se pueden utilizar para acceder a la base de datos UCCX directamente con el fin de extraer información para las aplicaciones de cartón, Quality Management, Workforce Management e informes históricos personalizados.

Aquí se describe la información del usuario, los permisos de cada usuario y el propósito deseado de cada usuario:

- **uccxhruser** - Este usuario tiene permisos de selección para muchas tablas de configuración e históricas en la base de datos UCCX y sólo se debe utilizar para informes históricos personalizados y Cisco Unified Workforce Management (WFM). Las consultas y los procedimientos almacenados ejecutados por este usuario pueden realizar consultas complejas y de larga duración. Debido al perfil de un usuario típico de informes históricos o WFM, estas consultas y procedimientos almacenados no deben ejecutarse con frecuencia, como ocurriría con una aplicación de paneles de pantalla.

  Aunque muchas aplicaciones de paneles de pared requieren datos incluidos en la

configuración y en las tablas históricas a las que tiene acceso uccxhruser, técnicamente no se admite el uso de este usuario para ejecutar consultas complejas y frecuentes con la base de datos UCCX a efectos de una aplicación de tablero de anuncios.
- **ucxstaff** - El usuario uccxstaff tiene acceso a las tablas Equipo, Recurso y Supervisor y se debe utilizar para Cisco Unified Quality Management (QM). Workforce Management debe utilizar uccxhruser ya que requiere acceso a tablas de datos históricos a las que el usuario uccxstaff no puede acceder.
- **uccxwallboard** - Este usuario sólo tiene permisos de selección en las tablas de base de datos en tiempo real que contienen instantáneas de estadísticas en tiempo real escritas desde la memoria del UCCX Engine. Los permisos de selección restringidos a las tablas RTCSQsSummary y RTICDStatistics significan que el usuario de uccxwallboard se debe utilizar para consultar la base de datos de UCCX con frecuencia con consultas simples y no complejas que se deben originar en una aplicación de pared.

Metodología de solución de problemas

En UCCX Release 10.0 y posteriores, ingrese el comando **utils uccx database dbperf** start <totalHours> <interval> para comenzar el seguimiento del rendimiento en la base de datos de **UCCX**. El argumento **interval** en este comando determina la periodicidad de la colección de seguimiento y el argumento **totalHours** determina la cantidad total de tiempo que el seguimiento se ejecuta antes de que se inhabilite. Estos parámetros son opcionales. Si no se especifican cuando se ejecuta el comando, se utilizan los valores predeterminados de 20 minutos y 10 horas.

Por ejemplo, ingrese el comando **utils uccx database dbperf start 24 30** para habilitar el seguimiento del rendimiento en la base de datos y recopilar datos sobre estadísticas de rendimiento cada 30 minutos durante 24 horas.

Las instrucciones para recopilar los datos obtenidos por el comando CLI se imprimen en el resultado del comando.

```
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:
```

Después de las **horas** totales proporcionadas, la recopilación de datos se detiene automáticamente. Para detener manualmente la recolección de datos, ingrese el comando **utils uccx database dbperf stop**.

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

Si la versión UCCX es la versión 9.0(2) o anterior y la **base de datos uccx utils dbperf** no está disponible, póngase en contacto con el Centro de asistencia técnica (TAC) para obtener más ayuda.

TAC ejecutará manualmente la secuencia de comandos dbperf.sh adjunta a la ID de bug de Cisco CSCuc68413 con acceso a cuenta de soporte remoto.

Cuando determina cuándo iniciar la ejecución de la secuencia de comandos manualmente o a través del comando CLI, la periodicidad y el tiempo total, asegúrese de que la CPU sea consumida por **uccxoninit** el proceso fluctúa significativamente o permanece alto durante esos períodos para reunir la información necesaria para el análisis de la causa raíz.

Además, ingrese periódicamente el comando **show process load** para determinar cuándo fluctúa la CPU para correlacionar los registros recolectados por el script de seguimiento dbperf.

Análisis de datos

Los registros recolectados por la ejecución del script dbperf de **onstat -g ses** 0 muestran las consultas activas que se emiten contra la base de datos **UCCX**. El uso elevado de la CPU en el proceso **uccxoninit** es típicamente el resultado de consultas complejas que toman mucho tiempo en ejecutarse. El objetivo es determinar las consultas que consumen más recursos, determinar el cliente de origen para esas consultas, deshabilitar las consultas del cliente para la resolución inmediata y optimizar las consultas de larga duración para la resolución permanente.

En los registros recolectados por el script dbperf, busque las consultas que probablemente causen altas fluctuaciones en la CPU o un alto consumo sostenido de la CPU por el proceso **uccxoninit**.

Consultas sospechosas:

- Se emiten desde sesiones conectadas como **uccxhruser** - Como se ha descrito anteriormente, **uccxhruser** tiene privilegios para seleccionar información de un gran número de tablas de configuración e históricas. Como resultado, se pueden crear consultas complejas y de larga duración en varias tablas y pueden tener un impacto en el rendimiento de la base de datos UCCX. Aunque no es absoluto, los usuarios de **uccxwallboard** y **uccxstaff** tienen un acceso tan limitado a las tablas dentro de la base de datos de UCCX, es poco probable que se produzcan consultas complejas que causen un impacto en el rendimiento emitido por estos usuarios. Además, las consultas emitidas por ucchrcare emitidas por UCCX Historical Reporting Client (HRC) o Cisco Unified Intelligence Center (CUIC) contra la base de datos de UCCX. Estas consultas son estáticas y no se pueden modificar y las consultas, junto con los índices pertinentes, se han escrito, probado y ajustado para lograr un impacto mínimo en el rendimiento.
- Realizar consultas intensivas en tablas históricas: las consultas que requieren que la base de datos UCCX realice varias uniones entre tablas, seleccione cantidades significativas de información o funcione en campos no indexados podrían causar un impacto en el rendimiento de la base de datos UCCX.

Aquí se muestra un ejemplo con una consulta compleja que implica una tabla RR. HH. ejecutada como **uccxhruser**:

```
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
```

El ejemplo anterior muestra una consulta compleja, ingresada por **uccxhruser** originada en el **WBBOX** host que podría causar un impacto en el rendimiento de la base de datos de UCCX si se ingresó a menudo o se ingresó periódicamente antes de que la consulta anterior hubiera devuelto los resultados.

Aunque poco frecuente, el rendimiento de la base de datos UCCX también puede degradarse (y el uso de la CPU de la **uccxoninit** el proceso fluctúa o permanece alto), como resultado del proceso de purga integrado. El proceso de depuración está diseñado para eliminar datos de la configuración y de las tablas históricas de la base de datos UCCX a fin de mantener el tamaño de la base de datos. La depuración se puede programar en función del tamaño de la base de datos o del registro más antiguo contenido en la base de datos.

Cuando se ejecuta el proceso de depuración, los datos se eliminan con una consulta. No se realiza iterativamente en función de la cantidad de registros que se deben eliminar. Esto significa que si la depuración detecta una gran cantidad de datos que deben eliminarse, emite una sola consulta en un intento de eliminar estos datos.

La modificación de la programación de depuración o de los parámetros de la página UCCX AppAdmin para programar la depuración para eliminar una gran cantidad de datos puede hacer que esta consulta única, en la siguiente depuración programada, tarde una cantidad significativa de tiempo en completarse. Por lo tanto, impulsa el uso de la CPU de la instancia de la base de datos.

En el resultado del script dbperf, se puede ver la consulta de depuración. Debe ser la única consulta ingresada por el usuario **uccxuser** que llama al **sp _purge** procedimiento almacenado.

```
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))));
```

Problemas Comunes

Basados en la experiencia reciente de Cisco TAC y Cisco Development Engineering, estos son los problemas más frecuentes que causan una alta utilización de la CPU en el proceso **uccxoninit**:

- Un cliente de la empresa se conecta como **uccxhruser** y ejecuta consultas complejas frecuentes en las tablas de paneles (RTICDStatistics y RTCSQsSummary) unidas a las tablas históricas para proporcionar una solución de informes personalizada o de cartón de pared. Para el uso de cartón, utilice solamente el usuario **uccxwallboard** y limite las consultas a las tablas en tiempo real. No se admite la posibilidad de consultar las tablas históricas o de configuración desde un panel o con una frecuencia similar a un panel de pared.
- Un cliente intenta ejecutar informes históricos personalizados en el nodo primario activo en lugar del nodo secundario. Sólo ejecute procedimientos almacenados, ya sean personalizados o predeterminados, que produzcan informes históricos en el nodo en espera. CUIC y HRC ejecutan consultas en el nodo en espera de forma predeterminada, pero al desarrollar un informe histórico personalizado, el desarrollador tiene la opción de qué nodo ejecutar estas consultas o ejecutar estos procedimientos almacenados.
- Cisco Workforce Management (WFM) emite una consulta compleja en la tabla ContactRoutingDetail para intentar filtrar en el campo startdatetime. De forma predeterminada, no se crea ningún índice en este campo de esta tabla, por lo que el rendimiento de esta consulta es deficiente. WFM ejecuta esta consulta periódicamente en un intento de sincronizar los datos de UCCX a WFM. Este problema se captura en el Id. de bug Cisco CSCtz23710 y se resuelve en la Versión 9.0(1)SR4 de WFM. Los clientes que experimenten este problema deberían actualizar a una versión de WFM que contenga una corrección para el Id. de error de Cisco CSCtz23710.
- Los umbrales de depuración se modifican para que la siguiente depuración programada intente eliminar una gran cantidad de datos. En lugar de modificar significativamente los parámetros de depuración en una sola actualización, las modificaciones de la programación de depuración se realizan de forma iterativa, con unos días entre las modificaciones de la configuración de depuración. Esto permite que el proceso de depuración elimine conjuntos más pequeños de datos en cada paso, lo que mejora el rendimiento de la operación de eliminación.
- La tabla DialingList es extremadamente grande. La tabla ListaMarcación almacena todos los contactos cargados en Campañas salientes. En las versiones 8.0 y 8.5 de UCCX, después de cargar millones de registros en las campañas salientes, se generan problemas de rendimiento y se consulta la tabla (lo que provoca un uso excesivo de la CPU en el proceso uccxoninit y una carga lenta de la página AppAdmin). Para mitigar los problemas de rendimiento, abra un caso TAC para la instalación de un script cron job que limpia la tabla DialingList. En la versión 9.0 de UCCX, se agregó un índice a esta tabla para realizar consultas más eficaces desde AppAdmin en un intento de mejorar el rendimiento. Este cambio resolvió el problema en todos los casos menos en los más extremos. En la versión 10.0 de UCCX, la lista de marcación se ha dividido en dos tablas, una para los contactos activos y otra para los contactos históricos, lo que proporciona una solución completa para este problema.