Alta utilização da CPU Informix

Contents

Introduction

Este documento descreve como as atividades do Unified Contact Center Express (UCCX), que exigem acesso ao banco de dados UCCX local, podem ser executadas lentamente. Faz com que as páginas AppAdmin sejam carregadas lentamente, que as atualizações no AppAdmin demorem muito para serem aplicadas, que o atraso na resposta a uma consulta no quadro de avisos, que o Workforce Manager não possa consultar os dados UCCX e outros problemas de desempenho e estabilidade.

O comando **show process load**, inserido na CLI, mostra que o **uccxoninit** consome uma grande quantidade de CPU. O processo **uccxoninit** representa a instância do banco de dados UCCX Informix executada no servidor UCCX.

Contribuído por Sridhar Chandrasekharan, Ryan LaFountain e Ben Wollak, engenheiros do TAC da Cisco.

Informações do recurso

O mecanismo de banco de dados que suporta o aplicativo UCCX é o Informix da IBM. As informações de configuração e histórico adicionadas à página AppAdmin do UCCX e produzidas pelo aplicativo UCCX são armazenadas na instância do UCCX Informix.

O aplicativo UCCX oferece três usuários que podem ser usados para acessar o banco de dados UCCX diretamente a fim de extrair informações para fins de aplicativos de quadro de parede, Quality Management, Workforce Management e relatórios de histórico personalizados.

As informações do usuário, as permissões de cada usuário e a finalidade de cada usuário são descritas aqui:

- **uccxhruser** - Este usuário tem permissões selecionadas para muitas tabelas de configuração e histórico no banco de dados UCCX e deve ser usado somente para relatórios de histórico personalizados e Cisco Unified Workforce Management (WFM). Consultas e procedimentos armazenados executados por este usuário podem executar consultas complexas e de longa execução. Devido ao perfil de um usuário típico de relatórios históricos ou WFM, essas consultas e procedimentos armazenados não devem ser executados com frequência, como ocorria em um aplicativo de wallboard.

Embora muitas aplicações de wallboard exijam dados contidos nas tabelas de configuração e histórico às quais o uccxhruser tem acesso, tecnicamente não há suporte para usar esse usuário para executar consultas complexas e frequentes no banco de dados UCCX para fins de um aplicativo de wallboard.

- **força de trabalho do uccx** - O usuário do uccxworker tem acesso às tabelas Equipe, Recurso e Supervisor e deve ser usado para o Cisco Unified Quality Management (QM). O Workforce Management deve usar uccxhruser, pois requer acesso a tabelas de dados históricos que não são acessíveis pelo usuário uccxworker.
- **quadro** - Este usuário tem permissões selecionadas somente nas tabelas de banco de dados em tempo real que contêm snapshots de estatísticas em tempo real gravadas na memória do Mecanismo UCCX. As permissões de seleção restritas às tabelas Resumo e Estatísticas de RTICDS significam que o usuário do uccxwallboard deve ser usado para consultar o banco de dados do UCCX frequentemente com consultas simples e não complexas destinadas a serem originadas por um aplicativo de wallboard.

Metodologia de solução de problemas

No UCCX Versão 10.0 e posterior, insira o comando **utils uccx database dbperf start <totalHours> <interval>** para iniciar o rastreamento de desempenho no banco de dados UCCX. O argumento **interval** nesse comando determina a periodicidade da coleta de rastreamento e o argumento **totalHours** determina o tempo total que o rastreamento executa antes de ser desabilitado. Esses parâmetros são opcionais. Se não forem especificados quando o comando for executado, serão usados os valores padrão de 20 minutos e 10 horas.

Por exemplo, insira o comando **utils uccx database dbperf start 24 30** para habilitar o rastreamento de desempenho no banco de dados e coletar dados sobre estatísticas de desempenho a cada 30 minutos por 24 horas.

As instruções para coletar os dados obtidos pelo comando CLI são impressas na saída do 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:
```

Após o **total de horas** fornecido, a coleta de dados é interrompida automaticamente. Para interromper manualmente a coleta de dados, insira o comando **utils uccx database dbperf stop**.

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

Se a versão do UCCX for 9.0(2) ou anterior e a **utils uccx database dbperf** não está disponível, entre em contato com o Centro de Assistência Técnica (TAC) para obter assistência.

O TAC executará o script dbperf.sh anexado ao Cisco bug ID CSCuc68413 manualmente com acesso à Conta de Suporte Remoto.

Quando você determina quando iniciar a execução do script manualmente ou por meio do comando CLI, a periodicidade e o tempo total garantem a CPU consumida pelo **uccxoninit** o processo oscila significativamente ou permanece alto durante esses períodos para coletar as informações necessárias para a análise da causa raiz.

Além disso, insira periodicamente o comando **show process load** para determinar quando a CPU flutua para correlacionar os registros coletados pelo script de rastreamento dbperf.

Análise de dados

Os registros coletados pela execução do script dbperf de **onstat -g ses 0** exibem consultas ativas que são emitidas em relação ao banco de dados UCCX. A alta CPU no processo **uccxoninit** é geralmente o resultado de consultas complexas que levam muito tempo para serem executadas. O objetivo é determinar as consultas que consomem a maioria dos recursos, determinar o cliente de origem para essas consultas, desativar as consultas do cliente para resolução imediata e otimizar as consultas de longa duração para resolução permanente.

Nos registros coletados pelo script dbperf, procure consultas que provavelmente causam grandes flutuações na CPU ou alto consumo sustentado de CPU pelo processo **uccxoninit**.

Consultas suspeitas:

- São emitidos de sessões conectadas como **uccxhruser** - Como descrito anteriormente, **uccxhruser** tem privilégios para selecionar informações de um grande número de tabelas de configuração e históricas. Como resultado, consultas complexas e de longa execução em várias tabelas podem ser construídas e podem ter impacto no desempenho do banco de dados UCCX. Embora não sejam absolutos, os usuários do **uccxwallboard** e **uccxworker** têm acesso tão limitado às tabelas no banco de dados UCCX, é improvável que as consultas complexas que causam impacto no desempenho emitidas por esses usuários. Além disso, consultas emitidas pelo uccxhrcare emitidas pelo UCCX Historical Reporting Client (HRC) ou pelo Cisco Unified Intelligence Center (CUIC) contra o banco de dados do UCCX. Essas consultas são estáticas e não podem ser modificadas, e as consultas, juntamente com indicações relevantes, foram escritas, testadas e ajustadas para um impacto mínimo no desempenho.
- Executar consultas intensas em tabelas históricas - Consultas que exigem que o banco de dados UCCX execute várias associações em tabelas, selecione quantidades significativas de informações ou opere em campos não indexados podem causar impacto no desempenho do banco de dados UCCX.

Um exemplo com uma consulta complexa que envolve uma tabela de RH executada como **uccxhruser** é mostrado aqui:

```
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
```

O exemplo acima mostra uma consulta complexa, inserida pelo **uccxhruser** originada da **WBBOX do host, que pode causar impacto no desempenho do banco de dados do UCCX se ela foi inserida com frequência ou foi inserida periodicamente antes da consulta anterior ter retornado resultados.**

Embora raro, o desempenho do banco de dados UCCX também pode diminuir (e a utilização da CPU do **uccxoninit** o processo oscila ou permanece alto), como resultado do processo de limpeza incorporado. O processo de limpeza foi projetado para excluir dados da configuração e tabelas de histórico no banco de dados UCCX para manter o tamanho do banco de dados. A limpeza pode ser agendada com base no tamanho do banco de dados ou no registro mais antigo contido no banco de dados.

Quando o processo de limpeza é executado, os dados são removidos com uma consulta. Ele não é feito iterativamente com base na quantidade de registros a serem removidos. Isso significa que se a limpeza detectar uma grande quantidade de dados que deve ser removida, ela emitirá uma única consulta na tentativa de remover esses dados.

A modificação da agenda ou parâmetros de limpeza da página UCCX AppAdmin para agendar a limpeza para remover uma grande quantidade de dados pode fazer com que esta única consulta, na próxima limpeza agendada, leve um tempo significativo para ser concluída. Portanto, ele aumenta a utilização da CPU da instância do banco de dados.

Na saída do script dbperf, a consulta de limpeza pode ser vista. Deve ser a única consulta inserida pelo usuário **uccxuser** que chama o procedimento armazenado **sp_purge**.

```
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 comuns

Com base na recente experiência do Cisco TAC e da Cisco Development Engineering, estes são os problemas mais comumente vistos que causam alta utilização da CPU no processo **uccxoninit**:

- Um cliente na empresa se conecta como **uccxhruser** e executa consultas complexas frequentes nas tabelas de wallboard (RTICDStatistics e RTCSQsSummary) associadas às tabelas históricas para fornecer uma solução de relatório personalizada ou de wallboard. Para uso em wallboard, use somente o usuário **uccxwallboard** e limite consultas às tabelas em tempo real. Não há suporte para a capacidade de consultar tabelas históricas ou de configuração de um quadro de parede ou com frequência semelhante a um quadro de parede.
- Um cliente tenta executar relatórios de histórico personalizados no nó primário ativo em vez do nó secundário. Execute somente procedimentos armazenados, personalizados ou padrão, que produzem relatórios de histórico no nó de standby. O CUIC e o HRC executam consultas no nó de standby por padrão, mas ao desenvolver um relatório de histórico personalizado, o desenvolvedor tem uma opção sobre qual nó executar essas consultas ou executar esses procedimentos armazenados.
- O Cisco Workforce Management (WFM) emite uma consulta complexa na tabela ContactRoutingDetail para tentar filtrar no campo startdatetime. Por padrão, não foi criado nenhum índice neste campo nesta tabela, portanto o desempenho desta consulta é ruim. O WFM emite esta consulta periodicamente na tentativa de sincronizar dados do UCCX com o WFM. Esse problema é capturado no bug da Cisco ID CSCtz23710 e é resolvido no WFM versão 9.0(1)SR4. Os clientes que tiverem esse problema devem atualizar para uma versão do WFM que contenha uma correção para o ID de bug da Cisco CSCtz23710.
- Os limites de limpeza são modificados de modo que a próxima limpeza agendada tenta remover uma grande quantidade de dados. Em vez de modificar significativamente os parâmetros de limpeza em uma única atualização, as modificações da agenda de limpeza são feitas repetidamente, com alguns dias entre as modificações da configuração de limpeza. Isso permite que o processo de limpeza remova conjuntos menores de dados em cada passagem, o que melhora o desempenho da operação de exclusão.
- A tabela DialingList é extremamente grande. A tabela DialingList armazena todos os contatos carregados em campanhas de saída. Nas versões 8.0 e 8.5 do UCCX, depois que milhões de registros são carregados para campanhas de saída, problemas de desempenho resultam que a tabela é consultada (o que causa alta CPU no processo uccxoninit e carregamento lento da página do AppAdmin). Para atenuar os problemas de desempenho, abra um caso do TAC para a instalação de um script de trabalho cron que limpe a tabela da lista de discagem. No UCCX Versão 9.0, um índice foi adicionado a esta tabela para consultas mais eficazes do AppAdmin na tentativa de melhorar o desempenho. Essa mudança resolveu o problema em todos, exceto nos casos mais extremos. No UCCX Versão 10.0, a DialingList foi dividida em duas tabelas, uma para contatos ativos e outra para contatos históricos, o que fornece uma correção abrangente para esse problema.