Utilisation du CPU élevé d'Informix

Contenu

Introduction

Ce document décrit comment les activités d'Unified Contact Center Express (UCCX), qui exigent l'accès aux bases de données local UCCX, pourraient exécuter lentement. Il fait charger des pages d'AppAdmin lentement, des mises à jour à AppAdmin pour prendre un longtemps de prendre l'affect, un retard en réponse à une requête de panneau mural, le gestionnaire de personnel pour ne pouvoir pas questionner des données UCCX, et d'autres questions de stabilité et performances.

Le chargement de processus d'exposition de commande, écrit dans le CLI, prouve que l'**uccxoninit** consomme un grand nombre de CPU. Le processus d'**uccxoninit** représente l'exemple de base de données Informix UCCX qui fonctionne sur le serveur UCCX.

Les informations de caractéristique

L'engine de base de données qui prend en charge l'application UCCX est Informix d'IBM. La configuration et les informations historiques qui sont ajoutées à la page d'AppAdmin d'UCCX et sont produites par l'application UCCX est stockée dans l'exemple UCCX Informix.

L'application UCCX fournit trois utilisateurs qui peuvent être utilisés pour accéder à la base de données UCCX directement afin d'extraire les informations aux fins des applications de panneau mural, de la gestion de la qualité, de la Gestion de personnel, et du rapport historique fait sur commande.

Les informations utilisateur, les autorisations de chaque utilisateur, et le but visé de chaque utilisateur sont décrits ici :

- **uccxhruser** - Cet utilisateur a des autorisations choisies aux beaucoup configuration et tables historiques dans la base de données UCCX et devrait être utilisé seulement pour le rapport historique et la Gestion faits sur commande de Cisco Unified Workforce (WFM). Les requêtes et le stored procedures exécutés par cet utilisateur pourraient exécuter des requêtes complexes et longues. En raison du profil d'un rapport historique typique ou utilisateur WFM, ces requêtes et stored procedures ne devrait pas être exécuté fréquemment comme se

produirait pour une application de panneau mural.

Bien que beaucoup d'applications de panneau mural exigent des données contenues dans la configuration et les tables historiques auxquelles l'uccxhruser a accès, il n'est pas techniquement pris en charge pour utiliser cet utilisateur pour exécuter des requêtes complexes et fréquentes contre la base de données UCCX aux fins d'une application de panneau mural.

- **uccxworkforce** - L'utilisateur d'uccxworkforce a accès aux tables d'équipe, de ressource, et de superviseur et devrait être utilisé pour la gestion de la qualité de Cisco Unified (QM). La Gestion de personnel devrait utiliser l'uccxhruser pendant qu'elle a besoin de l'accès aux tables de données historiques qui ne sont pas accessibles par l'utilisateur d'uccxworkforce.
- **uccxwallboard** - Cet utilisateur a des autorisations choisies seulement sur les tables de base de données en temps réel qui contiennent des instantanés des statistiques en temps réel écrits de la mémoire de l'engine UCCX. Les autorisations choisies limitées aux tables RTCSQsSummary et RTICDStatistics signifient que l'utilisateur d'uccxwallboard devrait être utilisé pour questionner la base de données UCCX fréquemment avec des requêtes simples et non-complexes destinées pour être originaire par une application de panneau mural.

Méthodologie

Dans la version 10.0 et ultérieures UCCX, sélectionnez la commande de **<totalHours> de <interval> de début de dbperf de base de données d'uccx d'utils** afin de commencer le suivi de représentation sur la base de données UCCX. L'argument d'**intervalle** dans cette commande détermine la périodicité de la collecte de suivi et l'argument de **totalHours** détermine la quantité totale de temps où le suivi exécute avant qu'il soit désactivé. Ces paramètres sont facultatifs. S'ils ne sont pas spécifiés quand la commande est exécutée les valeurs par défaut de 20 minutes et 10 heures sont utilisées.

Par exemple, écrivez les **utils que le début 30 24 de dbperf de base de données d'uccx** commandent afin d'activer le suivi de représentation sur la base de données et collecter des données sur des statistiques de représentation toutes les 30 minutes pendant 24 heures.

Des instructions de collecter les données obtenues par la commande CLI est imprimées dans la sortie de commande.

Après les **totalHours** indiqués, la collecte des informations arrête automatiquement. Afin d'arrêter manuellement la collecte des informations, sélectionnez la commande d'**arrêt de dbperf de base de données d'uccx d'utils**.

Si la version UCCX est la version 9.0(2) ou plus tôt et la commande de **dbperf de base de données d'uccx d'utils** n'est pas disponible, entrez en contact avec le centre d'assistance technique (TAC) pour davantage d'assistance.

Le TAC exécutera le script de dbperf.sh relié à l'ID de bogue Cisco CSCuc68413 manuellement avec l'accès de compte de support à distance.

Quand vous déterminez quand commencer l'exécution de script ou manuellement ou par la commande CLI, la périodicité, et le temps total, s'assurent que la CPU consommée par le processus d'**uccxoninit** flotte de manière significative ou reste élevée au cours de ces périodes afin de collecter les informations nécessaires pour l'analyse de cause principale.

Supplémentaire, sélectionnez périodiquement la commande de **chargement de processus d'exposition** de déterminer quand la CPU flotte afin de corréler les logs collectés par le script de suivi de dbperf.

Analyse de données

Les logs collectés par l'exécution du script de dbperf de l'**onstat - ses g 0** requêtes d'active d'exposition qui sont émises contre la base de données UCCX. La CPU de haute sur le processus d'**uccxoninit** est typiquement le résultat des requêtes complexes qui prennent un longtemps d'exécuter. Le but est de déterminer les requêtes qui consomment les la plupart des ressources, déterminent le client de source pour ces requêtes, désactivent les requêtes du client pour la résolution immédiate, et optimisent les requêtes longues pour la résolution permanente.

Dans les logs collectés par le script de dbperf, recherchez les requêtes que les fluctuations élevées de cause le plus susceptible dans la CPU ou la consommation élevée soutenue CPU par l'**uccxoninit** traitent.

Requêtes de suspect :

- Sont émis des sessions connectées comme **uccxhruser** - comme décrit plus tôt, l'**uccxhruser** a des privilèges de sélectionner les informations hors d'un gigantesque nombre de configuration et de tables historiques. En conséquence, le complexe, des requêtes longues à travers de plusieurs tables peut être construit et peut avoir des incidences des performances sur la base de données UCCX. Bien que non absolu, l'**uccxwallboard** et les utilisateurs d'**uccxworkforce** ont un tel accès limité aux tables dans la base de données UCCX, les requêtes de complexe qui entraînent l'incidence des performances émise par ces utilisateurs sont peu probables. Supplémentaire, des requêtes émises par l'**uccxhrc** sont émises par l'UCCX Historical Reporting Client (HRC) ou le centre d'intelligence de Cisco Unified (CUIC) contre la base de données UCCX. Ces requêtes sont statiques et ne peuvent pas être modifiées et les requêtes, avec des indicies appropriés, avoir été écrites, testées, et accordées pour l'incidence des performances minimale.
- Exécutez les requêtes intensives sur les tables historiques - Les requêtes qui exigent de la base de données UCCX d'exécuter multiple se joint à travers des tables, des quantités significatives choisies des informations ou actionne les champs en fonction non-répertoriés pourraient entraîner des incidences des performances à la base de données UCCX.

Un exemple avec une requête complexe qui implique une table heure exécutée comme **uccxhruser** est affiché ici :

```
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
```

L'exemple ci-dessus affiche une requête complexe, écrite par l'**uccxhruser** originaire de l'hôte **WBBOX** qui pourrait entraîner l'incidence des performances sur la base de données UCCX si elle était écrite souvent ou était écrite périodiquement avant que la requête précédente ait renvoyé des résultats.

Bien que rare, la représentation de base de données UCCX peut également dégrader (et l'utilisation du processeur du processus d'**uccxoninit** flotte ou demeure élevée), en raison du processus intégré de purge. Le processus de purge est conçu pour supprimer des données de la configuration et des tables historiques dans la base de données UCCX afin de mettre à jour la taille de la base de données. La purge peut être programmée a basé sur la taille de la base de données ou de l'enregistrement le plus ancien contenu dans la base de données.

Quand le processus de purge fonctionne, les données sont enlevées avec une requête. Il n'est pas fait itérativement basé sur la quantité d'enregistrements pour retirer. Ceci signifie que si la purge détecte un grand nombre de données qui doivent être enlevées, il émet une requête simple afin d'essayer d'enlever ces données.

La modification du programme ou des paramètres de purge de la page UCCX AppAdmin afin de programmer la purge pour enlever un grand nombre de données peut faire pour prendre cette requête simple, sur la purge ensuite programmée, une importante quantité de temps de se terminer. Par conséquent, il pilote vers le haut de l'utilisation du processeur de l'exemple de base de données.

Dans la sortie du script de dbperf, la requête de purge peut être vue. Ce devrait être la seule requête écrite par l'**uccxuser d'**utilisateur qui appelle la procédure stockée de **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))));
```

Problèmes courants

Basé sur Cisco récent TAC et l'expérience d'ingénierie de développement de Cisco, ce sont le plus généralement - les questions vues qui entraînent l'utilisation du CPU élevé sur le processus d'**uccxoninit** :

- Un client à l'entreprise se connecte comme **uccxhruser** et exécute des requêtes complexes fréquentes sur les tables de panneau mural (RTICDStatistics et RTCSQsSummary) jointes

avec les tables historiques afin de fournir un panneau mural ou une solution d'enregistrement de coutume. Pour l'usage de panneau mural, utilisez seulement les requêtes d'utilisateur et de limite d'**uccxwallboard aux** tables en temps réel. La capacité de questionner les tables de configuration historiques ou d'un panneau mural ou avec la fréquence semblable à un panneau mural n'est pas prise en charge.

- Tentatives d'un client d'exécuter des états historiques faits sur commande sur le noeud principal actif au lieu du noeud secondaire. Exécutez seulement le stored procedures, coutume ou par défaut, qui produisent des états historiques sur le noeud principal de non-engine. CUIC et HRC exécutent des requêtes sur le noeud de non-maître par défaut, mais quand développant un état historique fait sur commande, le développeur a un choix sur lequel noeud pour exécuter ces requêtes ou pour exécuter ce stored procedures.
- La Gestion de personnel de Cisco (WFM) émet une requête complexe sur la table de ContactRoutingDetail afin de tenter de filtrer sur le champ de startdatetime. Aucun index n'est créé sur ce champ dans cette table par défaut, ainsi la représentation de cette requête est pauvre. WFM émet cette requête périodiquement afin d'essayer de synchroniser des données d'UCCX à WFM. Cette question est capturée dans l'ID de bogue Cisco CSCtz23710 et est résolue dans la release 9.0(1)SR4 WFM. Les clients qui éprouvent cette question devraient améliorer à une version de WFM qui contient une difficulté pour l'ID de bogue Cisco CSCtz23710.
- Des seuils de purge sont modifiés tels que les tentatives ensuite programmées de purge d'enlever un grand nombre de données. Plutôt que modifiez de manière significative les paramètres de purge dans une mise à jour simple, les modifications de programme de purge sont faits itérativement, avec quelques jours entre les modifications de la configuration de purge. Ceci permet au processus de purge pour enlever de plus petits ensembles de données dans chaque passage, qui améliore la représentation de l'exécution d'effacement.
- La table de DialingList est extrêmement grande. La table de DialingList enregistre tous les contacts téléchargés aux campagnes sortantes. Dans des versions 8.0 et 8.5 UCCX, après que des millions d'enregistrements soient téléchargés aux campagnes en partance, les problèmes de performance résultent alors la table est questionnés (qui entraîne la CPU de haute sur le processus d'uccxoninit et le chargement lent de page d'AppAdmin). Afin d'atténuer les problèmes de performance, ouvrez une valise TAC pour l'installation d'un script du travail de cron qui nettoie la table de DialingList. Dans la version 9.0 UCCX, un index a été ajouté à cette table pour des requêtes plus efficaces d'AppAdmin afin d'essayer d'améliorer la représentation. Cette modification a résolu la question en tout mais les la plupart des cas extrêmes. Dans la version 10.0 UCCX le DialingList a été coupé en deux tables, une pour les contacts actifs et une autre pour les contacts historiques, qui fournissent une difficulté complète pour cette question.