Utilisation élevée du processeur Informix

Contenu

Introduction

Ce document décrit comment les activités Unified Contact Center Express (UCCX), qui nécessitent un accès local à la base de données UCCX, peuvent se dérouler lentement. Cela entraîne un chargement lent des pages AppAdmin, des mises à jour de AppAdmin prenant beaucoup de temps pour prendre effet, un retard dans la réponse à une requête de fond d'écran, Workforce Manager ne peut pas interroger les données UCCX, et d'autres problèmes de performances et de stabilité.

La commande **show process load**, entrée dans la CLI, montre que **uccxoninit** consomme une grande quantité de CPU. Le processus **uccxoninit** représente l'instance de base de données UCCX Informix qui s'exécute sur le serveur UCCX.

Contribué par Sridhar Chandrasekharan, Ryan LaFountain et Ben Wollak, ingénieurs du centre d'assistance technique de Cisco.

Informations sur les fonctionnalités

Le moteur de base de données qui prend en charge l'application UCCX est Informix d'IBM. Les informations de configuration et d'historique ajoutées à la page AppAdmin d'UCCX et produites par l'application UCCX sont stockées dans l'instance UCCX Informix.

L'application UCCX fournit trois utilisateurs qui peuvent être utilisés pour accéder à la base de données UCCX directement afin d'extraire des informations à des fins d'applications de fond d'écran, de gestion de la qualité, de gestion de la main-d'oeuvre et de rapports historiques personnalisés.

Les informations sur l'utilisateur, les autorisations de chaque utilisateur et la fonction prévue de chaque utilisateur sont décrites ici :

- **uccxhruser** - Cet utilisateur dispose d'autorisations de sélection pour de nombreuses tables de configuration et d'historique dans la base de données UCCX et doit être utilisé uniquement pour les rapports historiques personnalisés et Cisco Unified Workforce Management (WFM). Les requêtes et les procédures stockées exécutées par cet utilisateur peuvent exécuter des requêtes complexes et longues. En raison du profil d'un utilisateur WFM ou d'un rapport historique type, ces requêtes et procédures stockées ne doivent pas être exécutées fréquemment comme cela se produit pour une application de fond d'écran.

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

- **uccxstaff** - L'utilisateur uccxstaff a accès aux tables Team, Resource et Supervisor et doit être utilisé pour Cisco Unified Quality Management (QM). Workforce Management doit utiliser uccxhruser car il nécessite un accès aux tables de données historiques qui ne sont pas accessibles par l'utilisateur uccxstaff.
- **uccxwallboard** - Cet utilisateur dispose d'autorisations de sélection uniquement sur les tables de base de données en temps réel qui contiennent des instantanés de statistiques en temps réel écrites à partir de la mémoire du moteur UCCX. Les autorisations de sélection restreintes aux tables RTCSQsSummary et RTICDStatistics signifient que l'utilisateur uccxwallboard doit être utilisé pour interroger UCCX fréquemmentavec des requêtes simples et non complexes destinées à être obtenues par une application wallboard.

Méthodologie de dépannage

Dans UCCX version 10.0 et ultérieure, entrez la commande **utils uccx database dbperf start <totalHours> <interval>** afin de commencer le suivi des performances sur la base de données UCCX. L'argument **interval** de cette commande détermine la périodicité de la collection de traces et l'argument **totalHours** détermine la durée totale d'exécution du suivi avant sa désactivation. Ces paramètres sont facultatifs. Si elles ne sont pas spécifiées lors de l'exécution de la commande, les valeurs par défaut de 20 minutes et 10 heures sont utilisées.

Par exemple, entrez la commande **utils uccx database dbperf start 24 30** afin d'activer le suivi des performances sur la base de données et de collecter des données sur les statistiques de performance toutes les 30 minutes pendant 24 heures.

Les instructions de collecte des données obtenues par la commande CLI sont imprimées dans le résultat de la commande.

```
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:
```

Après le **totalHeures** donné, la collecte de données s'arrête automatiquement. Afin d'arrêter manuellement la collecte de données, entrez la commande **utils uccx database dbperf stop**.

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

Si la version UCCX est la version 9.0(2) ou antérieure et **utils uccx base de données dbperf** n'est pas disponible, contactez le Centre d'assistance technique (TAC) pour obtenir de l'aide.

TAC exécutera le script dbperf.sh associé à l'ID de bogue Cisco CSCuc68413 manuellement avec l'accès au compte d'assistance à distance.

Lorsque vous déterminez quand démarrer l'exécution du script manuellement ou via la commande CLI, la périodicité et le temps total, assurez-vous que le CPU consommé par le **uccxoninit** le processus fluctue considérablement ou reste élevé au cours de ces périodes afin de recueillir les informations nécessaires à l'analyse des causes premières.

En outre, entrez périodiquement la commande **show process load** pour déterminer quand le CPU fluctue afin de corréler les journaux collectés par le script de traçage dbperf.

Analyse des données

Les journaux collectés par le script dbperf lors de l'exécution d'**onstat -g ses 0** montrent les requêtes actives émises sur la base de données UCCX. Le CPU élevé sur le processus **uccxoninit** est généralement le résultat de requêtes complexes qui prennent beaucoup de temps à s'exécuter. L'objectif est de déterminer les requêtes qui consomment le plus de ressources, de déterminer le client source pour ces requêtes, de désactiver les requêtes du client pour une résolution immédiate et d'optimiser les requêtes de résolution permanente en cours d'exécution.

Dans les journaux collectés par le script dbperf, recherchez les requêtes qui provoquent le plus probablement de fortes fluctuations du CPU ou une consommation élevée de CPU soutenue par le processus **uccxoninit**.

Requêtes suspectes :

- Sont émises à partir de sessions connectées en tant que **uccxhruser** - Comme décrit précédemment, **uccxhruser** dispose de privilèges permettant de sélectionner des informations dans un grand nombre de tables de configuration et d'historiques. Par conséquent, des requêtes complexes et longues sur plusieurs tables peuvent être créées et peuvent avoir des effets sur les performances de la base de données UCCX. Bien qu'ils ne soient pas absolus, les utilisateurs de **uccxwallboard** et **uccxworkers** ont un accès si limité aux tables de la base de données UCCX, les requêtes complexes qui ont un impact sur les performances émises par ces utilisateurs sont peu probables. En outre, les requêtes émises par uccxhrcare par l'UCCX Historical Reporting Client (HRC) ou Cisco Unified Intelligence Center (CUIC) sur la base de données UCCX. Ces requêtes sont statiques et ne peuvent pas être modifiées et les requêtes, ainsi que les index pertinents, ont été écrites, testées et réglées pour un impact minimal sur les performances.
- Effectuer des requêtes intensives sur des tables historiques : les requêtes qui nécessitent que la base de données UCCX effectue plusieurs jointures entre les tables, sélectionne des quantités importantes d'informations ou opère sur des champs non indexés peuvent avoir des

répercussions sur les performances de la base de données UCCX.

Un exemple avec une requête complexe qui implique une exécution d'une table HR sous la forme **uccxhruser** est présenté 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 montre une requête complexe, entrée par **uccxhruser** provenant de l'hôte **WBBOX qui pourrait avoir un impact sur les performances de la base de données UCCX si elle a été saisie souvent ou régulièrement avant que la requête précédente n'ait renvoyé les résultats.**

Bien que rare, les performances de la base de données UCCX peuvent également se dégrader (et l'utilisation du CPU du **uccxoninit** fluctue ou reste élevé), en raison du processus de purge intégré. Le processus de purge est conçu pour supprimer des données des tables de configuration et d'historique de la base de données UCCX afin de maintenir la taille de la base de données. La purge peut être planifiée en fonction de la taille de la base de données ou du plus ancien enregistrement contenu dans la base de données.

Lorsque le processus de purge s'exécute, les données sont supprimées avec une requête. Il n'est pas effectué itérativement en fonction de la quantité d'enregistrements à supprimer. Cela signifie que si la purge détecte une grande quantité de données qui doit être supprimée, elle émet une requête unique pour tenter de supprimer ces données.

La modification de la planification ou des paramètres de purge de la page UCCX AppAdmin afin de planifier la purge pour supprimer une grande quantité de données peut entraîner la fin de cette requête unique, lors de la purge planifiée suivante, pour un temps important. Par conséquent, il augmente l'utilisation du CPU de l'instance de base de données.

Dans la sortie du script dbperf, la requête de purge peut être vue. Il doit s'agir de la seule requête entrée par l'utilisateur **uccxuser** qui appelle la procédure stockée **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

Sur la base de l'expérience récente du centre d'assistance technique de Cisco et de l'ingénierie de développement de Cisco, voici les problèmes les plus courants qui provoquent une utilisation élevée du CPU sur le processus **uccxoninit** :

- Un client de l'entreprise se connecte en tant qu'**utilisateur uccxhruser** et exécute des requêtes complexes fréquentes sur les tables de fond d'écran (RTICDStatistics et RTCSQsSummary) jointes aux tables historiques afin de fournir une solution de fond d'écran ou de rapport personnalisé. Pour l'utilisation du fond d'écran, utilisez uniquement l'utilisateur **uccxwallboard** et limitez les requêtes aux tables en temps réel. La possibilité d'interroger les tables d'historique ou de configuration à partir d'un fond d'écran ou avec une fréquence similaire à celle d'un fond d'écran n'est pas prise en charge.
- Un client tente d'exécuter des rapports historiques personnalisés sur le noeud principal actif au lieu du noeud secondaire. Exécutez uniquement les procédures stockées, personnalisées ou par défaut, qui produisent des rapports historiques sur le noeud de secours. CUIC et HRC exécutent les requêtes sur le noeud de secours par défaut, mais lors du développement d'un rapport historique personnalisé, le développeur a le choix du noeud sur lequel exécuter ces requêtes ou exécuter ces procédures stockées.
- Cisco Workforce Management (WFM) émet une requête complexe sur la table ContactRoutingDetail afin de tenter de filtrer sur le champ startdatetime. Aucun index n'est créé par défaut sur ce champ de cette table, de sorte que les performances de cette requête sont médiocres. WFM émet régulièrement cette requête pour tenter de synchroniser les données d'UCCX vers WFM. Ce problème est capturé dans l'ID de bogue Cisco CSCtz23710 et est résolu dans WFM version 9.0(1)SR4. Les clients qui rencontrent ce problème doivent passer à une version de WFM qui contient un correctif pour l'ID de bogue Cisco CSCtz23710.
- Les seuils de purge sont modifiés de sorte que la purge planifiée suivante tente de supprimer une grande quantité de données. Plutôt que de modifier de manière significative les paramètres de purge dans une seule mise à jour, les modifications de planification de purge sont effectuées itérativement, avec quelques jours entre les modifications de configuration de purge. Cela permet au processus de purge de supprimer des ensembles de données plus petits dans chaque passe, ce qui améliore les performances de l'opération de suppression.
- La table DialingList est extrêmement grande. La table DialingList stocke tous les contacts téléchargés vers les campagnes sortantes. Dans les versions 8.0 et 8.5 d'UCCX, une fois des millions d'enregistrements téléchargés vers les campagnes sortantes, des problèmes de performances surviennent, puis la table est interrogée (ce qui entraîne une utilisation élevée du CPU dans le processus uccxoninit et un chargement lent de la page AppAdmin). Afin d'atténuer les problèmes de performances, ouvrez un dossier TAC pour l'installation d'un script de travail cron qui nettoie la table DialingList. Dans UCCX version 9.0, un index a été ajouté à cette table pour des requêtes plus efficaces d'AppAdmin afin d'améliorer les performances. Ce changement a résolu le problème dans tous les cas sauf les plus extrêmes. Dans UCCX version 10.0, la liste de numérotation a été divisée en deux tables, l'une pour les contacts actifs et l'autre pour les contacts historiques, ce qui fournit une solution complète à

ce problème.