Informix High CPU 사용률

목차

소개

이 문서에서는 로컬 UCCX 데이터베이스 액세스가 필요한 UCCX(Unified Contact Center Express) 활동이 느리게 수행되는 방법에 대해 설명합니다.이로 인해 AppAdmin 페이지가 느리게 로드되고, AppAdmin에 대한 업데이트가 영향을 받는 데 오랜 시간이 걸리며, 웰보드 쿼리에 대한 응답이 지연되며, Workforce Manager가 UCCX 데이터를 쿼리할 수 없고 기타 성능 및 안정성 문제가 발생합니다.

CLI에 입력된 **show process load** 명령은 uccxoninit가 많은 CPU를 사용한다는 것을 보여줍니다 .uccxoninit **프로세스**는 UCCX 서버에서 실행되는 UCCX Informix 데이터베이스 인스턴스를 **나타냅니다**.

기고자: Sridhar Chandrasekharan, Ryan LaFountain, Ben Wollak, Cisco TAC 엔지니어

기능 정보

UCCX 애플리케이션을 지원하는 데이터베이스 엔진은 IBM의 Informix입니다.UCCX의 AppAdmin 페이지에 추가되고 UCCX 애플리케이션에서 생성한 컨피그레이션 및 기록 정보는 UCCXInformix 인스턴스에 저장됩니다.

UCCX 애플리케이션은 웰보드 애플리케이션, 품질 관리, 인력 관리 및 맞춤형 내역 보고를 위해 정보를 추출하기 위해 UCCX 데이터베이스에 직접 액세스하는 데 사용할 수 있는 세 명의 사용자를 제공합니다.

사용자 정보, 각 사용자의 권한 및 각 사용자의 용도는 다음과 같습니다.

- **uccxhruser** - 이 사용자는 UCCX 데이터베이스의 여러 컨피그레이션 및 기록 테이블에 대한 선택 권한을 가지며 사용자 지정 내역 보고 및 Cisco WFM(Unified Workforce Management)에만 사용해야 합니다. 이 사용자가 실행하는 쿼리 및 저장 프로시저는 복잡하고 오래 실행되는 쿼리를 수행할 수 있습니다.일반적인 기록 보고 또는 WFM 사용자의 프로필으로 인해 이러한 쿼리 및 저장 프로시저가 웰보드 응용 프로그램에 대해 자주 실행되지 않아야 합니다.

  많은 웰보드 애플리케이션에서 ucxhruser가 액세스할 수 있는 컨피그레이션 및 기록 테이블 내에 포함된 데이터가 필요하지만, 웰보드 애플리케이션을 위해 이 사용자를 사용하여 UCCXdatabase에 대해 복잡하고 빈번한 쿼리를 실행하는 것은 기술적으로 지원되지 않습니다 .
- **uccxworkforce** - uccxworkforce 사용자는 팀, 리소스 및 수퍼바이저 테이블에 액세스할 수 있으

며 Cisco QM(Unified Quality Management)에 사용해야 합니다. Workforce Management는 uccxworkforce 사용자가 액세스할 수 없는 기록 데이터 테이블에 액세스해야 하므로 uccxhruser를 사용해야 합니다.

- **uccxwallboard** - 이 사용자는 UCCX 엔진의 메모리에서 작성된 실시간 통계의 스냅숏을 포함하는 실시간 데이터베이스 테이블에서만 사용 권한을 선택할 수 있습니다.테이블 RTCSQsSummary 및 RTICStstatistics로 제한된 선택 권한은 uccxwallboard 사용자가 월보드 응용 프로그램에서 소싱하도록 의도된 간단한 비복합 쿼리를 사용하여 UCCX 데이터베이스를 자주 쿼리하는 데 사용되어야 함을 의미합니다.

문제 해결 방법론

UCCX 릴리스 10.0 이상에서는 UCCX 데이터베이스의 성능 추적을 시작하려면 **utils uccx 데이터베이스 dbperf start <totalHours> <interval>** 명령을 입력합니다.이 명령의 **interval** 인수는 추적 컬렉션의 주기성을 결정하고 **totalHours** 인수는 추적을 비활성화하기 전에 실행하는 총 시간을 결정합니다.이러한 매개변수는 선택 사항입니다.명령을 실행할 때 이 값을 지정하지 않으면 기본값인 20분 및 10시간이 사용됩니다.

예를 들어, 데이터베이스에서 성능 추적을 활성화하고 24시간 동안 30분마다 성능 통계에 대한 데이터를 수집하려면 utils **uccx database dbperf start 24 30** 명령을 입력합니다.

CLI 명령에서 얻은 데이터를 수집하는 지침은 명령 출력에 출력됩니다.

```
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:
```

지정된 **totalHours**가 지나면 데이터 수집이 자동으로 중지됩니다.데이터 수집을 수동으로 중지하려면 utils **uccx 데이터베이스 dbperf stop 명령을** 입력합니다.

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

UCCX 버전이 Release 9.0(2) 이전 버전인 경우 **utils uccx 데이터베이스 dbperf 데이터베이스** 명령을 사용할 수 없습니다. 자세한 내용은 TAC(Technical Assistance Center)에 문의하십시오.

TAC는 Cisco 버그 ID CSCuc68413에 첨부된 dbperf.sh 스크립트를 원격 지원 계정 액세스로 수동으로 실행합니다.

수동으로 또는 CLI 명령을 통해 스크립트 실행을 시작할 시기를 결정하는 경우 주기 및 총 시간을 확인합니다. **uccxoninit** 근본 원인 분석에 필요한 정보를 수집하기 위해 이 기간 동안 프로세스가 크

게 변하거나 계속 높게 유지됩니다.

또한 **show process load** 명령을 주기적으로 입력하여 dbperf 추적 스크립트에서 수집한 로그를 상호 연결하기 위해 CPU가 변동하는 시기를 확인합니다.

데이터 분석

dbperf 스크립트의 onstat -g **ses 0**에서 수집한 로그는 UCCX 데이터베이스에 대해 실행된 활성 쿼리를 표시합니다.ucxoninit 프로세스의 높은 CPU**는** 일반적으로 실행하는 데 시간이 오래 걸리는 복잡한 쿼리의 결과입니다.목표는 리소스를 가장 많이 사용하는 쿼리를 결정하고, 해당 쿼리에 대한 소스 클라이언트를 결정하고, 즉각적인 해결을 위해 클라이언트의 쿼리를 비활성화하고, 영구적인 해결을 위해 장기 실행 쿼리를 최적화하는 것입니다.

dbperf 스크립트에서 수집한 로그에서 CPU의 고변이나 uccxoninit 프로세스에 의한 지속적인 고 CPU 소비를 유발할 수 있는 **쿼리를** 찾습니다.

의심스러운 쿼리:

- uccxhruser로 연결된 세션**에서** 발급됨 - 앞에서 설명한 대로 uccxhruser는 방대한 수의 컨피그레이션 및 기록 테이블에서 정보를 선택할 수 있는 권한을 가집니다.따라서 여러 테이블에 걸쳐 복잡한 장기 실행 쿼리를 생성할 수 있으며 UCCX 데이터베이스에 성능 영향을 줄 수 있습니다.uccxwallboard **및 uccxworkforce** 사용자는 UCCX 데이터베이스 내의 테이블에 대한 액세스가 제한적이므로 이러한 사용자가 발행한 성능에 영향을 주는 복잡한 쿼리가 발생할 가능성은 거의 없습니다.또한 UCCX 데이터베이스에 대해 UCCX Historical Reporting Client(HRC) 또는 Cisco CUIC(Unified Intelligence Center)에서 발급한 uccxhrcare 쿼리가 UCCX 데이터베이스에 대해 발행합니다.이러한 쿼리는 정적이며 수정할 수 없으며 관련 지표와 함께 성능에 미치는 영향을 최소화하도록 작성, 테스트 및 조정된 것입니다.
- 기록 테이블에 대해 집중적인 쿼리 수행 - UCCX 데이터베이스가 테이블 간에 여러 조인을 수행해야 하는 쿼리, 상당한 양의 정보를 선택하거나 인덱싱되지 않은 필드에서 작동하면 UCCX 데이터베이스에 성능 영향을 줄 수 있습니다.

HR 테이블이 uccxhruser로 실행되는 복잡한 쿼리**의 예**는 다음과 같습니다.

```
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
```

위의 예는 호스트 WBBOX에서 **소싱된 uccxhruser가 입력한** 복합 쿼리를 보여 줍니다. 이 쿼리를 자주 입력했거나 이전 쿼리가 결과를 반환하기 전에 정기적으로 입력한 경우 UCCX 데이터베이스에 성능 영향을 줄 수 있습니다.

드물기는 하지만 UCCX 데이터베이스 성능도 저하될 수 있으며 **uccxoninit** 기본 삭제 프로세스의 결과로 프로세스가 변화하거나 높게 유지됩니다.비우기 프로세스는 UCCX 데이터베이스 내의 컨

피그레이션 및 기록 테이블에서 데이터베이스 크기를 유지하기 위해 데이터를 삭제하도록 설계되었습니다.데이터베이스 크기 또는 데이터베이스에 포함된 가장 오래된 레코드를 기준으로 비우기 일정을 잡을 수 있습니다.

삭제 프로세스가 실행되면 하나의 쿼리로 데이터가 제거됩니다.제거할 레코드 양에 따라 반복적으로 수행되지 않습니다.즉, 삭제에서 제거해야 하는 많은 양의 데이터를 탐지하면 이 데이터를 제거하기 위해 단일 쿼리를 실행합니다.

비우기 일정을 잡기 위해 UCCXAppAdmin 페이지에서 비우기 일정 또는 매개변수를 수정하면 다음 예정된 비우기 시 이 단일 쿼리가 완료되는 데 상당한 시간이 걸릴 수 있습니다.따라서 데이터베이스 인스턴스의 CPU 사용률을 높입니다.

dbperf 스크립트의 출력에서 비우기 쿼리를 볼 수 있습니다.**sp_**purge 저장 프로시저를 호출하는 **uccxuser**가 입력한 유일한 쿼리여야 합니다.

```
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))));
```

일반적인 문제

최근 Cisco TAC 및 Cisco Development Engineering 경험을 기반으로 uccxoninit 프로세스에서 CPU 사용률이 높은 가장 일반적으로 나타나는 **문제입니다**.

- 엔터프라이즈의 클라이언트는 uccxhruser로 **연결되며**, 월보드 또는 사용자 지정 보고 솔루션을 제공하기 위해 기록 테이블과 조인된 월보드 테이블(RTICDStatics 및 RTCSQsSummary)에서 자주 복잡한 쿼리를 실행합니다.월보드 사용의 경우 uccxwallboard 사용자만 사용하고 쿼리를 실시간 테이블로 제한합니다.월보드에서 또는 월보드와 유사한 빈도로 기록 또는 구성 테이블을 쿼리하는 기능은 지원되지 않습니다.
- 클라이언트는 보조 노드 대신 활성 기본 노드에서 사용자 지정 기록 보고서를 실행하려고 시도합니다.대기 노드에서 기록 보고서를 생성하는 저장 프로시저(사용자 지정 또는 기본값)만 실행합니다.CUIC 및 HRC는 기본적으로 대기 노드에서 쿼리를 실행하지만 사용자 지정 기록 보고서를 개발할 때 개발자는 이러한 쿼리를 실행하거나 이러한 저장 프로시저를 실행할 노드를 선택할 수 있습니다.
- Cisco WFM(Workforce Management)은 startdatetime 필드에서 필터링하려고 시도하기 위해 ContactRoutingDetail 테이블에 복잡한 쿼리를 실행합니다.기본적으로 이 테이블의 이 필드에 인덱스가 만들어지지 않으므로 이 쿼리의 성능이 저하됩니다.WFM은 UCCX에서 WFM으로 데

이터를 동기화하기 위해 정기적으로 이 쿼리를 실행합니다.이 문제는 Cisco 버그 ID CSCtz23710에서 캡처되고 WFM 릴리스 9.0(1)SR4에서 해결됩니다. 이 문제가 발생한 고객은 Cisco 버그 ID CSCtz23710에 대한 수정 사항이 포함된 WFM 버전으로 업그레이드해야 합니다 .

- 다음 예약된 비우기가 대량의 데이터를 제거하려고 시도하도록 비우기 임계값이 수정됩니다 .단일 업데이트에서 비우기 매개변수를 크게 수정하지 않고 비우기 구성 수정 며칠 사이에 비우기 일정을 반복적으로 수정합니다.이렇게 하면 삭제 프로세스가 각 단계에서 더 작은 데이터 집합을 제거하므로 삭제 작업의 성능이 향상됩니다.
- DialingList 테이블이 매우 큽니다.DialingList 테이블은 아웃바운드 캠페인에 업로드된 모든 연락처를 저장합니다.UCCX 릴리스 8.0 및 8.5에서는 수백만 개의 레코드가 아웃바운드 캠페인에 업로드된 후 성능 문제가 발생하면 테이블이 쿼리됩니다. 그러면 uccxoninit 프로세스에서 CPU가 높고 AppAdmin 페이지 로드가 느려집니다. 성능 문제를 완화하려면 DialingList 테이블을 정리하는 cron 작업 스크립트 설치를 위한 TAC 케이스를 엽니다.UCCX Release 9.0에서는 성능 향상을 위해 AppAdmin의 보다 효과적인 쿼리를 위해 인덱스가 이 테이블에 추가되었습니다.이 변화는 가장 극단적인 경우를 제외하고 모든 문제를 해결했습니다.UCCX Release 10.0에서는 DialingList가 활성 연락처용 테이블과 기록 연락처용 두 개의 테이블로 분할되어 이 문제에 대한 포괄적인 수정 사항을 제공합니다.