記事
Tomoko Furuzono · 2020年9月17日 17m read

SQLクエリパフォーマンスの監視

Caché 2017以降のSQLエンジンには新しい統計一式が含まれています。
これらの統計は、クエリの実行回数とその実行所要時間を記録します。

これは、多くのSQLステートメントを含むアプリケーションのパフォーマンスを監視する人や最適化を試みる人にとっては宝物のような機能ですが、一部の人々が望むほどデータにアクセスするのは簡単ではありません。

この記事と関連するサンプルコードでは、このような情報の使用方法と、日次統計の概要を定期的に抽出してアプリケーションのSQLパフォーマンス履歴記録を保持する方法について説明します。

※詳細については、下記ドキュメントページもご参考になさってください。
https://docs.intersystems.com/iris20201/csp/docbook/DocBook.UI.Page.cls?KEY=GSQLOPTsqlstmts

記録内容

SQLステートメントが実行されるたびに、所要時間が記録されます。
この処理は非常に軽量であり、オフにすることはできません。
コストを最小限に抑えるため、統計はメモリに保持されてから定期的にディスクに書き込まれます。
このデータには当日にクエリが実行された回数と、その平均所要時間と合計所要時間が含まれます。

データはすぐにはディスクに書き込まれません。ただし、統計はデータが書き込まれた後に通常は1時間ごとに実行されるようにスケジュールされている「SQLクエリ統計の更新」タスクによって更新されます。 このタスクは手動で起動できますが、クエリをテスト中にリアルタイムで統計を表示したい場合は、プロセス全体の実行時間が若干長くなります。

警告：InterSystems IRIS 2019以前では、**%Studio.Project:Deploy**
メカニズムを使用して**配置されたクラスやルーチン**
に埋め込まれたSQLのこれらの統計は収集されません。 サンプルコードを破壊するものは何もありませんが、コストがかかるものが何も表示されないため、すべてが正常であると思い込んでしまう可能性があります（私はそう思い込んでしまいました）。

確認できる情報

クエリのリストを管理ポータルで確認できます。 [SQL] ページに移動し、[SQLステートメント] タブをクリックしてください。 これは実行中および調査中の新しいクエリには適していますが、膨大な数のクエリが実行中の場合は管理しきれなくなる可能性があります。

別の方法として、SQLを使用してクエリを検索することができます。
情報は、INFORMATIONSCHEMAスキーマのテーブルに保存されます。 このスキーマには多数のテーブルが含まれていますが、この記事の最後にいくつかのサンプルSQLクエリを掲載しています。

統計が削除されるタイミング

クエリのデータは、クエリが再コンパイルされるたびに削除されます。
そのため、動的クエリの場合はキャッシュ済みのクエリがパージされる可能性があります。
埋め込みSQLの場合、それはSQLが埋め込まれているクラスやルーチンが再コンパイルされるタイミングになります 。

※IRIS2020.1
以降では、埋め込みSQLも動的クエリと同様に、クエリが再コンパイルされるたびにデータが削除されます。

稼働中のサイトでは1日以上統計が保持されると思われますが、統計を保持しているテーブルは、レポートや長期分析を実行するための長期参照ソースには使用できません。

情報を要約するには

パフォーマンスレポートを生成する際は、作業しやすい永続的なテーブルに毎晩データを抽出することをお勧めします。 クラスが日中にコンパイルされる場合は一部の情報が失われる可能性がありますが、それによってスロークエリの分析に実質的な違いが出ることはほとんどありません。

以下のコードは、各クエリの日次概要に統計を抽出する方法の例です。
このコードには3つの短いクラスが含まれています。

- 毎晩実行する必要があるタスク。
- DRL.MonitorSQLは、INFORMATIONSCHEMAテーブルからデータを抽出して保存するメインクラスです。
- 3番目のクラスであるDRL.MonitorSQLTextは（長い可能性がある）クエリテキストを1回だけ保存し、毎日の統計にクエリのハッシュのみを保存する最適化です。

サンプルに関する注意

このタスクは前日分のデータを抽出するため、午前0時を過ぎた直後にスケジュールする必要があります。
履歴データが存在する場合は、それをエクスポートできます。
過去120日間を抽出するには以下を実行します。
Do ##class(DRL.MonitorSQL).Capture($h-120,$h-1)

最も古いバージョンの統計ではDateがSQLに公開されていなかったため、サンプルコードでは^rIndexグローバルを直接読み取っています。
私が含めたバリエーションはインスタンス内のすべての名前空間を通ってループしますが、それが常に適切であるとは限りません。

抽出されたデータに対してクエリを実行するには

データを抽出したあと、以下のクエリを実行することで最も重いクエリを見つけることができます。

```
SELECT top 20
S.RunDate,S.RoutineName,S.TotalHits,S.SumpTIme,S.Hash,t.QueryText
from DRL.MonitorSQL S
left join DRL.MonitorSQLText T on S.Hash=T.Hash
where RunDate='08/25/2019'
order by SumpTime desc
```

また、高コストなクエリのハッシュを選択すると、そのクエリの履歴を表示できます。

```
SELECT S.RunDate,S.RoutineName,S.TotalHits,S.SumpTIme,S.Hash,t.QueryText
from DRL.MonitorSQL S
left join DRL.MonitorSQLText T on S.Hash=T.Hash
where S.Hash='CgOlfRw7pGL4tYbiijYznQ84kmQ='
order by RunDate
```

私は今年の初めに本番サイトからデータを収集し、最も高コストなクエリを調べました。 1つのクエリの平均実行時間は6秒未満でしたが、1日あたり14,000回呼び出されており、毎日合計で24時間近くかかっていました。 事実上、この1つのクエリで1つのコアが完全に占有されていました。
さらに悪いことに、1時間かかる2番目のクエリは1番目のクエリのバリエーションでした。

RunDate	RoutineName	Total Hits	Total Time	Hash	Query
03/16/2019		14,576	85,094	5xDSguu4PvK04se2pPiOexeh6aE=	DECLARE C CURS SELECT * INTO :% :%col(3) , :%col(4)
03/16/2019		15,552	3,326	rCQX+CKPwFR9zOplmtMhxVnQxyw=	DECLARE C CURS SELECT * INTO :% :%col(3) , :%col(4)
03/16/2019		16,892	597	yW3catzQzC0KE9euvIJ + o4mDwKc =	DECLARE C CURS SELECT * INTO :% :%col(3) , :%col(4) :%col(6) , :%col(7)
03/16/2019		16,664	436	giShyiqNR3K6pZEt7RWAcen55rs=	DECLARE C CURS SELECT * , TKGR :%col(1) , :%col(2)
03/16/2019		74,550	342	4ZClMPqMfyje4m9Wed0NJzxz9qw=	DECLARE C CURS SELECT …

表1：顧客サイトでの実際の結果

INFORMATIONSCHEMA スキーマのテーブル

このスキーマのテーブルは統計だけでなく、使用されているクエリ、カラム、インデックスなどを追跡しています。 通常はそのSQLステートメントが最初のテーブルになり、"Statements.Hash = OtherTable.Statement" のような形で結合されます。

これらのテーブルに直接アクセスし、1日で最も高コストなクエリを見つけるための対応するクエリは以下のようになります。

```
SELECT DS.Day,Loc.Location,DS.StatCount,DS.StatTotal,S.Statement,S.Hash
FROM INFORMATIONSCHEMA.STATEMENTDAILYSTATS DS
left join INFORMATIONSCHEMA.STATEMENTS S
on S.Hash=DS.Statement
left join INFORMATIONSCHEMA.STATEMENTLOCATIONS Loc
on S.Hash=Loc.Statement
where Day='08/26/2019'
order by DS.stattotal desc
```

より体系的なプロセスのセットアップを検討しているかどうかに関係なく、SQLを使用する大規模アプリケーションを持つすべての方に今すぐこのクエリを実行することをお勧めします。

特定のクエリが高コストであると表示される場合は、次を実行して履歴を取得できます。

```
SELECT DS.Day,Loc.Location,DS.StatCount,DS.StatTotal,S.Statement,S.Hash
FROM INFORMATIONSCHEMA.STATEMENTDAILYSTATS DS
left join INFORMATIONSCHEMA.STATEMENTS S
on S.Hash=DS.Statement
left join INFORMATIONSCHEMA.STATEMENTLOCATIONS Loc
on S.Hash=Loc.Statement
where S.Hash='jDqCKaksff/4up7Ob0UXlkT2xKY='
order by DS.Day
```

日次統計を週出するためのコードサンプル

#SQL #パフォーマンス #監視 #Caché #InterSystems IRIS

ソースURL:
https://jp.community.intersystems.com/post/sql%E3%82%AF%E3%82%A8%E3%83%AA%E3%83%91%E3%83%95%E3%82%A9%E3%83%BC%E3%83%9E%E3%83%B3%E3%82%B9%E3%81%AE%E7%9B%A3%E8%A6%96