文章
Hao Ma · 一月 10, 2021 阅读大约需 15 分钟

InterSystems 最佳实践系列---APM – 监控 SQL 查询性能

自 Caché 2017 以后，SQL 引擎包含了一些新的统计信息。
这些统计信息记录了执行查询的次数以及运行查询所花费的时间。

对于想要对包含许多 SQL
语句的应用程序的性能进行监控和尝试优化的人来说，这是一座宝库，但访问数据并不像一些人希望的那么容易。

本文和相关的示例代码说明了如何使用这些信息，以及如何例行提取每日统计信息的摘要，并保存应用程序的 SQL 性能的历史记录。

记录了什么？

每次执行 SQL 语句时，都记录花费的时间。 这是非常轻量的操作，无法关闭。
为了最大程度地降低开销，统计信息保留在内存中并定期写入磁盘。
数据包括一天中执行查询的次数以及所花费的平均时间和总时间。

数据不会立即写入磁盘，并且在写入之后，统计信息将由 “ 更新 SQL
查询统计信息 ” 任务更新，该任务通常计划为每小时运行一次。
该任务可以手动触发，但是如果你希望在测试查询时实时查看统计信息，则整个过程需要一点耐心。

警告：在 InterSystems IRIS 2019 及更早版本中，不会针对已使用 **%Studio.Project:Deploy 机制部署**的**类或例程**中的嵌入式 SQL 收集这些统计信息。 示例代码不会有任何中断，但这可能会使你产生误导（我被误导过），让你以为一切正常，因为没有查询显示为高开销。

如何查看信息？

你可以在管理门户中查看查询列表。 转到 SQL 页面，点击 “ SQL 语句 ” 选项卡。
对于你正在运行并查看的新查询，这种方式很好；但是如果有数千条查询正在运行，则可能变得难以管理。

另一种方法是使用 SQL 搜索查询。 信息存储在 INFORMATION_SCHEMA 模式的表中。
该模式含有大量表，我在本文的最后附上了一些 SQL 查询示例。

何时删除统计信息？

每次重新编辑查询时会删除其数据。 因此对于动态查询，这可能意味着清除缓存的查询时。 对于嵌入式 SQL，则意味着重新编译在其中嵌入 SQL 的类或例程时。

在活跃的站点上，可以合理预期统计信息将保存超过一天，但是存放统计信息的表不能用作运行报告或长期分析的长期参考源。

如何汇总信息？

我建议每天晚上将数据提取到永久表中，这些表在生成性能报告时更易于使用。
如果在白天编译类，可能会丢失一些信息，但这不太可能对慢速查询的分析产生任何实际影响。

下面的代码示例说明了如何将每个查询的统计信息提取到每日汇总中。 它包括三个简短的类：

- 一个应在每晚运行的任务。
- DRL.MonitorSQL 是主类，用于从 INFORMATIONSCHEMA 表提取数据并存储。
- 第三个类 DRL.MonitorSQLText
  是一个优化类，它存储一次（可能很长的）查询文本，并且只将查询的哈希存储在每天的统计信息中。

示例说明

该任务提取前一天的信息，因此应安排在午夜后不久执行。

你可以导出更多历史数据，只要其存在。 要提取过去 120 天的数据

Do ##class(DRL.MonitorSQL).Capture($h-120,$h-1)

该示例代码直接读取全局 ^rIndex，因为最早版本的统计信息未将日期公开给 SQL。

我所包括的变体将循环实例中的所有命名空间，但这并不总是合适的。

如何查询已提取的数据

提取数据后，您可以通过运行以下语句查找最繁重的查询

```
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
```

此外，如果选择了开销大的查询的哈希，可以通过以下语句查看该查询的历史记录

```
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
```

今年早些时候，我获取了一个活跃站点的数据，然后查看了开销最大的查询。 有一个查询的平均时间不到 6 秒，但每天被调用 14000 次，加起来每天消耗的时间将近 24 小时。 实际上，一个核心完全被这个查询占用。
更糟糕的是，第二个查询要花一个小时，它是第一个查询的变体。

运行日期	例程名称	总命中次数	总时间	哈希	查询文本（有节略）
03/16/2019		14,576	85,094	5xDSguu4PvK04se2pPiOexeh6aE=	DECLARE C CURSOR FOR SELECT * INTO :%col(1) , :%col(2)

				, :%col(3) , :%col(4) …
03/16/2019	15,552	3,326	rCQX+CKPwFR9zOplmtMhxVnQxyw=	DECLARE C CURSOR FOR SELECT * INTO :%col(1) , :%col(2) , :%col(3) , :%col(4) , …
03/16/2019	16,892	597	yW3catzQzC0KE9euvIJ+o4mDwKc=	DECLARE C CURSOR FOR SELECT * INTO :%col(1) , :%col(2) , :%col(3) , :%col(4) , :%col(5) , :%col(6) , :%col(7) ,
03/16/2019	16,664	436	giShyiqNR3K6pZEt7RWAcen55rs=	DECLARE C CURSOR FOR SELECT * , TKGROUP INTO :%col(1) , :%col(2) , :%col(3) , ..
03/16/2019	74,550	342	4ZClMPqMfyje4m9Wed0NJzxz9qw=	DECLARE C CURSOR FOR SELECT …

表 1：客户站点的实际结果

INFORMATIONSCHEMA 模式中的表

除了统计信息外，此模式中的表还会跟踪查询、列、索引等的使用位置。 通常，SQL 语句是起始表，它的连接方式类似于 “ Statements.Hash=OtherTable.Statement ”。

直接访问这些表以查找一天中开销最大的查询，这一操作的等效查询是...

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

无论你是否考虑建立一个更系统的过程，我都建议每个使用 SQL 处理大型应用程序的人今天都运行这个查询。

如果某个特定查询显示为高开销，则可以通过运行以下语句获取历史记录

```
SELECT DS.Day,Loc.Location,DS.StatCount,DS.StatTotal,S.Statement,S.Hash
FROM INFORMATION_SCHEMA.STATEMENT_DAILY_STATS DS
left join INFORMATION_SCHEMA.STATEMENTS  S
on S.Hash=DS.Statement
left join INFORMATION_SCHEMA.STATEMENT_LOCATIONS  Loc
```

```
on S.Hash=Loc.Statement
where S.Hash='jDqCKaksff/4up7Ob0UXlkT2xKY='
order by DS.Day
```

每日提取统计信息的代码示例

#SQL #性能 #监视 #Caché #InterSystems IRIS

源URL: https://cn.community.intersystems.com/post/intersystems-%E6%9C%80%E4%BD%B3%E5%AE%9E%E8%B7%B5%E7%B3%BB%E5%88%97-apm-%E2%80%93-%E7%9B%91%E6%8E%A7-sql-%E6%9F%A5%E8%AF%A2%E6%80%A7%E8%83%BD