Nhibernate/Hibernate 常见性能问题—不规范事务

伴随这华通各个子项目的路上上线试运行。基于 Nh 的 Plinq 这套后台框架，接触的还算比较长，对其中 NH 的性能问题有一些皮毛的了解和思考。打算写出来与需要的人分享。不需要的请猛戳浏览器右上角的红 X。

而且关于 NH，公司不少项目再用。至于用的对与不对，也没有一个标准。

我花了不少时间阅读 Nh 的英文和中文资料。也会去看一些如 CodeProject 和 stackoverflow 知名技术社区论坛，查阅和学习 NH 不为人知或者被人忽视的是使用细节。

所以从本篇开始，我将写一个系列，将学习心得和成果写出来。来分析和探讨 NH 或者基于 NH 二次封装的各种 Orm 框架在使用的时候应该注意的性能问题。

有说的不对的或者不准确的，欢迎各种鸟和各种神吐槽指点。

之所以同时将 Hibernate 也一块纳入进来，是因为 Nh 是移植 Hibernate 的思想，只是平台不一样而已，可能有的表述在 Java 平台上并不准确，但是他们所存在的问题一定是相同或者相似的

废话不多说，来干货。

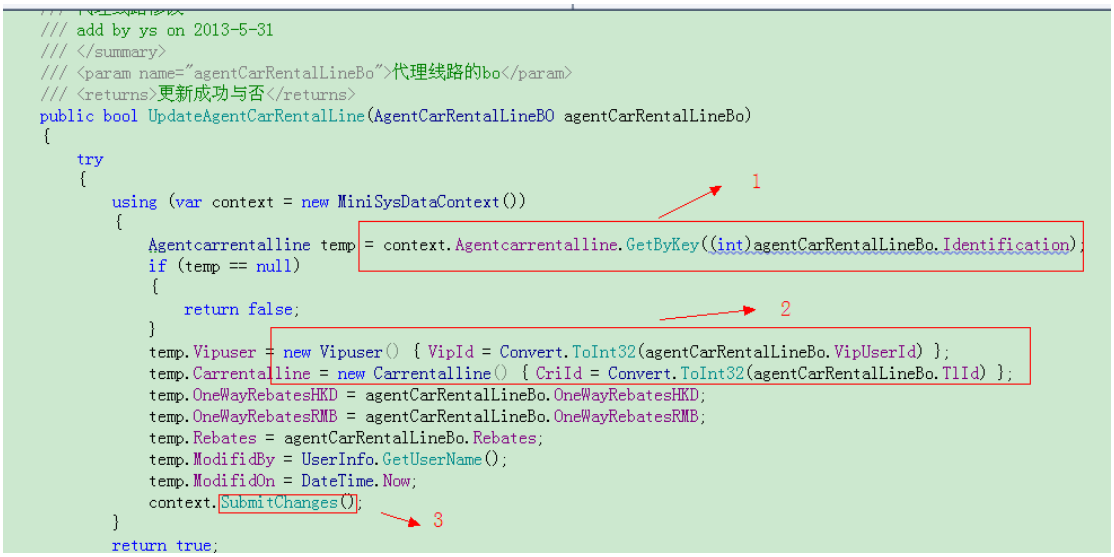
1. 不规范的使用事务
   很多人用过 Nh/Hibernate，很流畅很丝滑很好用。但是对于他们的内部机制可能并不是很熟悉。来说下，Nh/H 有一级和二级缓存之说。一级 Cache 为 Session 级缓存。一个有效 Session 内，数据是缓存共享的。二级缓存在 SessionFactory 级别，为所有 ISession 共享，可插拔配置。不多说。

   如果没有特殊说明，以下所有关于缓存的称呼均指一级 cache。

   只要用 Nh，我们就在和一级 cache 打交道。Nh 这套框架，设计者为了保证 cache 数据和数据库数据的一致性。Nh 所有数据库操作都是在事务中处理的。有更新删除插入，也包括查询。这种未经过显示声明使用的事务称为隐式事务。

   在 mssql 中，这种隐式的事务称作自动提交事务，autocommit transaction。

   有一个问题，当我们使用 nh 作为 orm 的时候，如果没有显示的声明使用事务，那么，nh 每一次进行 db 操作的时候将会自动采用数据库的隐式事务模式，每执行一个 sql，自动创建一个隐私事务。就像下面的



分析下问题。Using，有了一个 ISession，创建一个数据库操作实例。无问题
看 1，2，3，按照 nh 的设计，本处未显示的声明使用事务，那么这个 update 方法内，所有的数据库操作语句执行完毕会有几个事务呢。

回到隐式事务上来。隐私事务是一个 level 级别很高的事务级别。意味着，每一个 sql 代码段都会在自己独自的事务中运行,这种情况实际上事务粒度已经达到了很小，处于原子级了。一句 sql 就是一个执行单元，具有原子性，要么不执行，要么全部执行。关于隐私事务，如果开启了隐私事务，那么，所有的 db 操作将会在 sql 提交或者回滚后自动开启新事务，无法描述这种事务的开始，但是，我们可以指定事务的 commit 和 rollback。

假设上面的代码 2 有过更新。(2 处用法，执行过程是先 select，根据 counts 进行 insert 和 update 操作的，无疑问，wiki 上有专篇，自己找下 ok 了)

那么 3 处 submmit 的时候，交给数据库的 sql 会有四条(1 处 1 条 select,2 处会产生过四条 sql)，ok，在 submmitchange 的时候，按照上面的代码。每条 sql 都使用隐私事务。这个过程一共产生了 5 个事务。数据库会执行 5 次事务的创建和销毁过程。这会带来严重的性能消耗。第二个带来的问题是，直接导致数据库容错性降低。

你关注过这个么。

所以我在国外的一些社区看到”I wrote python”这种级别的大神，给的强烈建议是，在使用 Nh 作为 orm 框架的时候。任何数据库操作都应该显示的使用事务，即使是查询操作。因为查询这种操作，能降低事务隔离级别，确保我们不会读到脏数据。确保 Isession 跟踪到的数据与 db 中完全一致。

Nh 官方给出的建议是，任何数据库操作，包括查询，都应该在显示事务下运行。并且强烈反对在没有事务的情况下使用 ISession。

有一些伪性能控，会说，事务会带来性能损耗。好，来涨下姿势。
先将事务操作带来的安全性，这个特性放在一边不说，假定使用事务是有很大性能消耗

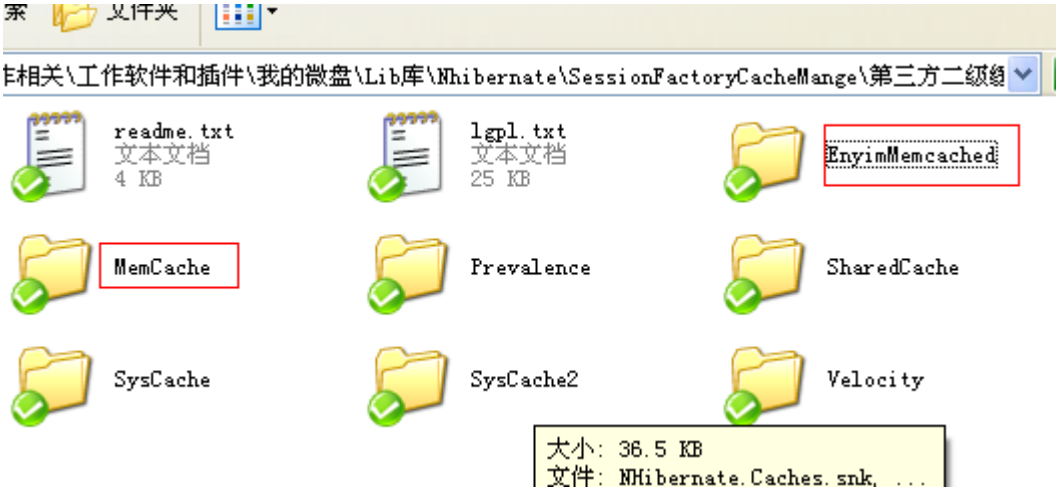
的，所以我们要对涉及到的事务操作进行一些优化处理，前面我们提到过的。使用 NH 时候，db 操作会使用隐私事务模式。数据库接收的 sql 始终在事务(隐私事务)下运行，，db 会执行这么多 sql，每一个 sql 都在事务下运行，请问你优化得过来么?无力吐槽

所以我们面临的真正问题是，究竟是一个 sql 语句段来创建一个事务呢，还是应该在使用批处理的使用使用一个事务。很显然的，聪明的你能想到，一个 sql 来创建一个事务所带来的性能损耗，自然比一个批处理创建一个事务高得多得多

说到这里。我想表达的意思是，其实，我们完全可以通过改变事务隔离级别，来控制使用事务的数量和所使用事务的锁级别。最普遍的办法就是，降低事务级别，一定程度上增大事务粒度。

实际上，NH 将 Commit 操作视为 ISession 跟踪的数据项和 DB 进行同步的一种操作单元。如果你没有显示的进行 commit 调用，NH 自己是不知道进行数据同步的。也许你使用了 Flush()这个方法，但是，很明显，你没有明白我前面讲的那么多的意思。Fush()操作直接进行数据同步，再涨下姿势。Flush 操作做了两个工作，清理缓存，执行 sql，不可避免的又使用了隐私事务。通常情况下。显示的调用 Flush 操作，则意味着，你没有正确的使用事务。明白了把??

好，再讲一下不显示的使用事务所带来的一个问题。上面我已经提到过 ，二级缓存的问题。二级缓存 NH 默认是没有启动的，而且这个东西有一些第三方的二级缓存组件，我收藏的，具体如下，懒得打字了。



比较有名的是 MemCache，我之前配过一次。难度不大，要学习的自己 google。跑题了。讲事务和二级缓存的关系。

如果配置了二级缓存，那么不显示的使用事务，将会直接影响二级缓存的数据同步。我看了一些相关资料，NH 为了尽力保证二级缓存和数据库中的数据一致，做了大量的处理。最关键的数据同步手段就是依赖于显示事务。

通俗点说。只有在事务提交的时候，二级缓存才会和数据库进行同步更新。

可能有人会说，隐私事务呢。像下面的这种用法

```
using(var session = sessionFactory.OpenSession())
{
    var post = session.Get<Post>(1);
}
```

这个查询将会直接导致二级缓存清空。结果是，这个查询会去数据库中查询，而没法从二级 cache 中获取数据。我曾经试过。不信的化可以写 demo，再 NH Profile 检测。

上面那种用法，当进行 insert 提交的时候，二级缓存组件是处于工作状态的。实际结果是，只有当二级缓存组件发现一个事务提交，那么，他才会去缓存数据项。否则，二级缓存组件，将不会缓存任何已经加载的数据项

所以说来说去，配了二级缓存，使用事务那就是更加必要的。正确用法是外层显示的使用声明事务，在配了二级 cache 后，强烈要求这点，否则你取的数据可能会有各种问题，或者没有，或者是脏的，那这种数据来操作，不敢想象有什么后果。
正确用法如下

```
using(var session = sessionFactory.OpenSession())
using(var tx = session.BeginTransaction())
{
    var post = session.Get<Post>(1);
    tx.Commit();
}
```

显示的声明事务，commit，这样用，二级缓存不会有问题。也是正确的用法。

所以呢。上面这段代码是有问题的。即使他它是我写的。但是我得承认它仍然有很大的问题。

2013-10-15