Nhibernate 性能问题 04—缓存过量访、数据库过量访问和 Session 过量产生

继上篇 Nhibernat03—臭名昭著的 N+1 问题一文中提到的，复杂 N+1 问题的延续。

本次继续 NH 中另一个非常常见的问题，缓存的过量访问和数据库的过量访问。尤其是后者，更为常见。

1. Session 级别的过量数据库访问

   计算机是执行速度非常快的，哪怕是人感受不到的 1 微秒，在计算机看来仍然是无比漫长。所以，在项目生产中。不可避免的要考虑到这种执行时间。如果你人为的都已经能感受到你的代码执行起来很慢很慢。那说明你的代码已经有了非常非常严重的问题。

   在业务实现的过程中，有几种耗时的操作，远程服务调用，跨线程访问，远程机器访问。无论是那种访问形式，记好一条，本地的的执行时间远远比远程的执行时间短，相差数百倍。

   描述一种场景，SOA 这种结构下。业务通过 API 接口来实现。也就是说，client 要获取数据，必须进行远程调用(Remote calling)，这种操作是非常耗时的，但是这种场景又是非常常见的。怎么办呢？

   选择合理的 API 解决方案。(保密性，轻量级的数据交换格式，易扩展，易维护，版本易管理易更替等)
   在合理的前提下，减少请求次数，减少数据库的访问次数。

   关于 api 方案的选择，不在本文讨论之列。我要说的是第二条。

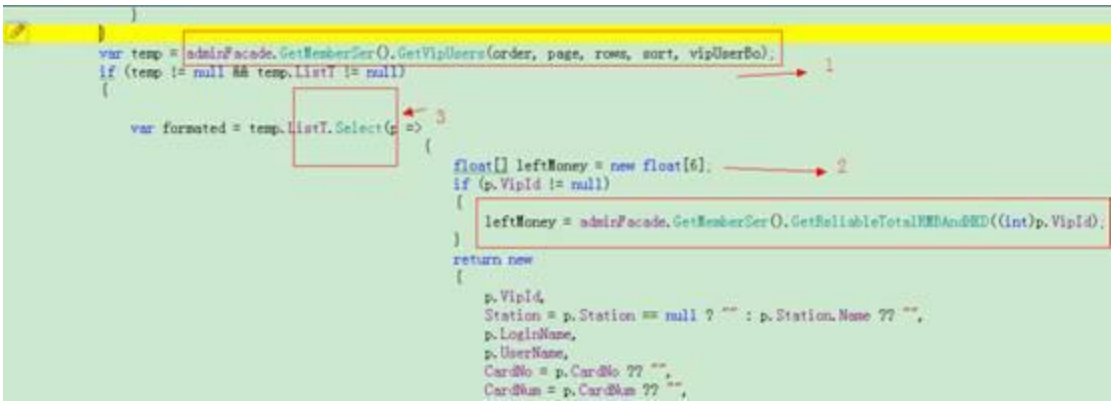
   开发中，访问数据库，无论你是读还是写。都是远程访问。所以要尽可能的减少这种访问的次数。

   那么数据库过量访问是怎么产生的呢?

   1. N+1 问题产生的大量查询语句
   2. 在循环中，访问数据库，这是很多人喜欢犯的毛病。
   3. 一次性的更新(插入或者删除)过多的对象，1w 个对象将会产生 1w 句操作语句。
   4. 在具体业务中，需要大量的查询才能得到我们需要的数据

   第一种情况，参考 N+1 问题的说明，=>N+1 问题

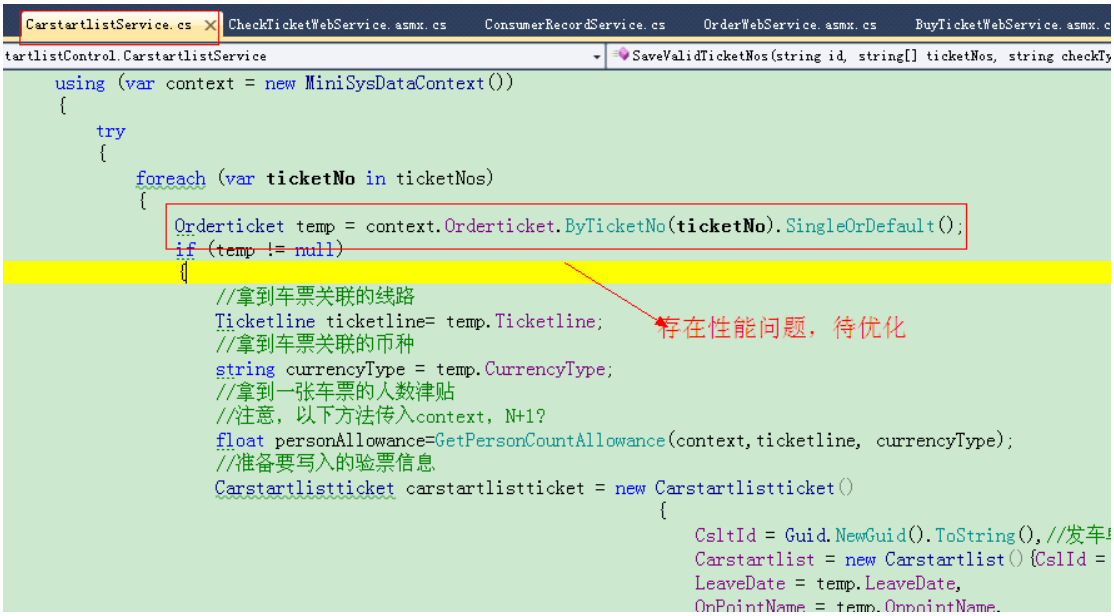
   第二种情况，在循环中访问数据库通常情况下完全是 bug，当封装的方法是独立的，而且每个方法内部都通过 using()这样打开一个数据库连接，然后获取数据。那简直是太坑爹了。完全的 N+1 问题，导致的结果就是过量数据库访问。看点图

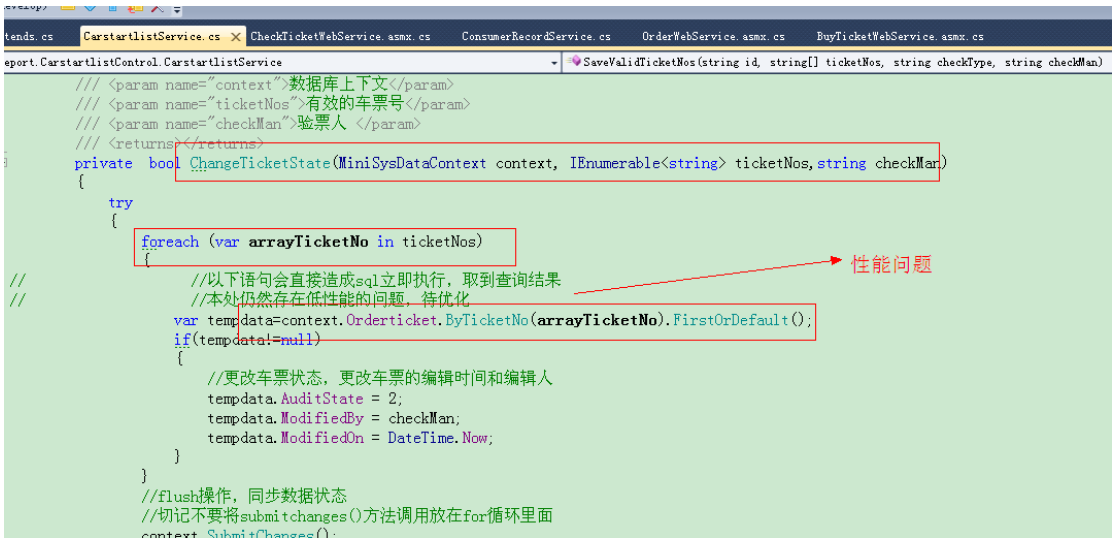


huatongmis 中，加载一个带余额的用户列表,loop 中有数据库访问。

好吧，我承认这些是我写的：(。这样的代码，一页几十条数据问题应该是不大的，但是如果页数有 500 甚至更多，呃，问题就很严重了。。。。。。有解决办法么?有，OneSessionPerRequest 模式，

再看两张图，都是循环中访问数据库的





第三种情况，操作大量对象。这个得使用批处理，如果是原生的 Nhibernate，设置 batch size 参数，限制 NH 一次给 DB 发送的 sql 语句，避免一次产生大量操作语句。

最后一种情况比较有意思，如果我们需要的数据来自不同的数据库，虽然我们现在使用的都是单库多表的方案，如果以后，有读写分离，有分库分表呢，数据来源自不同的数据库完全是很正常的。我们需要有多个查询来获得我们所需要的数据，既然每个查询都是独立分开的，第四种情况如果出现，前面三种情况都会伴随出现的。

如何避免第四种情况的问题呢，好再 Nhibernate 提供了 MultiQuery 和 MultiCriteria 查询接口来避免这种问题。MultiQuery 和 MultiCriteria 都可以将原来的多次独立查询合并成一次数据库访问查询。如果你打算采用这种方案来解决第四种情况下出现的问题。我建议深入研究下 MultiQuery 和 MultiCriteria 查询。

官方出的文档说这两种方案可以实现跨 DB 的查询。但是仅仅是说可以，具体效果和灵活性如何还不得而知。因为我没有试过。连 demo 都没有写过。

2. Cache 级别的过量访问

需要先声明下这个问题，这个 Cache 指的是分布式缓存。比如 Radis 和 Membercache 这种。部署在单独的缓存服务上的。

场景是这样的，用 Orm 取数据的时候，都是先从缓存服务器上取，而不是直接去 db 中国取，之前已经说过 DB 访问是远程访问调用，耗时是肯定的。而直接冲缓存中取，即使是跨机器的访问，速度还是远比访问远程数据库拿数据来的快。

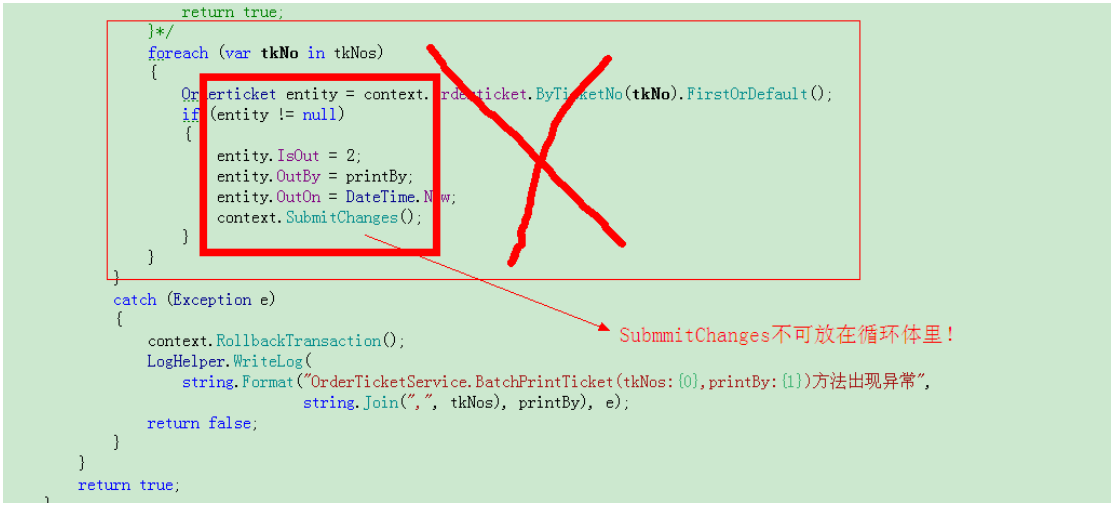
如果有分布式缓存做支撑，业务实现中又产生了大量的查询语句，去 Cache 服务器去查是必然的，查询语句多了肯定会出问题，所以，语句过多，直接会产生 Cache 的过量访问问题、

HT 项目中并没有使用分布式缓存。连 NH 的二级缓存都没有使用。所以，不存在此问题。相信未来短时间内，大家都不会碰到这种问题。有需要了再说。

3. 过量数据写入

严格来说，过量数据写入应该是 Session 级别过量 DB 访问的一种，之所以拿出来单独讲，是因为我觉得，这个问题 HT 已经碰到了，而且需要讲一下怎么解决。

先看个代码



在循环中使用 Submmitchanges。暂且不说在 loop 中有数据库访问的问题。

这里拿这端代码是想表达，过量数据写入的问题，把那个 Context.SubmmitChanges 方法拿到 loop 外面来

假设一种使用场景。还是华通的场景，下单买票，批量购票，一次性购票 500 或者 1000 张。循环中修改车票的状态，创建人，创建时间等，然后 loop 完了一次性 SubmmitChanges。会不会有问题呢?肯定有。

HT 用的是 mysql。那么 Nhibernate 提供的 Provider 一定是生成了 500 或者 1000 调 insert 语句，数据没问题，出问题是，Nhibernate 怎么将这些语句交给 DB 去执行。

我记得原生 Nhibernate 中是配置文件中是有 batch size 参数的，可配置，也可以使用 session.SetBatchSize()在运行时进行配置 HT 中用的 PlinqO-Nhibernate 是经过封装的 Nhibernate，不知道有没有在批处理上做处理。反正 batch size 的设置我没有找到代码。

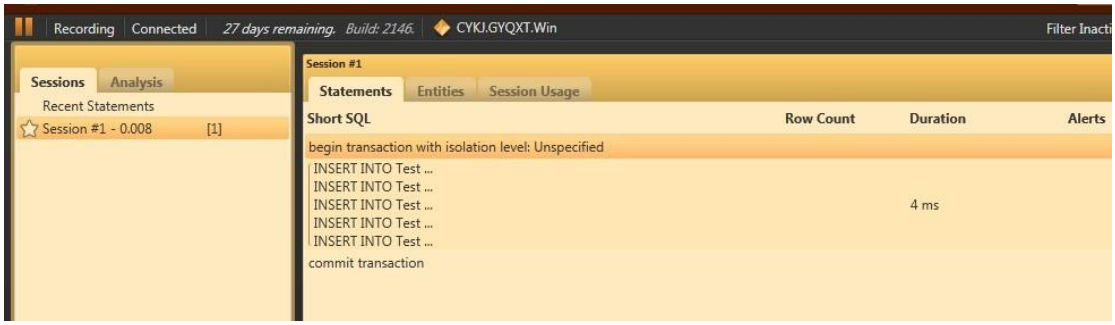
嗯。来看这个图

connection.isolation	ADO.NET事务隔离级别， 查看System.Data.IsolationLevel类来了解各个值的具体意义，但请注意 都不支持所有的隔离级别。 例如： Chaos, ReadCommitted, ReadUncommitted, RepeatableRead, Serializable, Unspe
connection.release_mode	指定ADO.NET何时释放数据库连接。 例如： auto (默认)	on_close	after_transaction 注意,这些设置仅对通过ISessionFactory.OpenSession得到的ISession起作用。 对于通过ISessionFactory.GetCurrentSession得到的ISession， 所配置的ICurrentSessionContext实现控制这些ISession的连接释放模式。
command_timeout	指定NHibernate生成的IDbCommands对象的超时时间。 .
adonet.batch_size	指定用ADO.Net的批量更新的数量，默认设置为0（不启用该功能）。
proxyfactory.factory_class	代理工厂配置。用于指定是哪个中间件提供的延迟加载代理

原生的 NH 是没有启用语句批处理的。不过 Plino NH 这东西我用 NH profile 监测发现，貌似一个 Session 内通常都是有多条语句的。

Batch size 的作用，是限制 sql 语句的产生，主要用于 Insert，update。避免一次产生太多的 sql 给数据库

在批量插入的时候，确实显示的一个 Session 有多条 Insert，就是不知道是不是 batch size。看下面这图



看了这张图，我有必要再提一下，事务的使用。具体参考 NH 性能问题 01-不规范事务

好，上面说的是，过量数据插入的问题，兴许有人要问，为什么不能将 Submmitchanges 放循环里。

对，为什么呢？Submmitchangge 方法，执行的是一次 DB 通信，他会将自上次同步以来，NH 内部 Session 跟踪的所有发生过改变的对象同步到 DB。前面提到，跟 DB 通信是远程访问，是非常耗时的。那么在循环中使用 Submmitchange，完全是超级耗时的操作。

另外一个，在 WebServer 内部，每一次的 submmit 都是一次状态同步。难道维护这些状态不需要内存空间么？难道对跟宗对象状态进行检查不需要时间么？

所以在循环体内用 Summmitchangge 是测底错误的做法、不然你卖鸡蛋为什么不一个个拿到商店去卖，还懂得用篮子装好一次性拿去卖。无力吐槽了:(

小结下，批量操作，要用显示事务；要设置 batch size，明确的告诉 Provider 使用批量处理；Submmit 操作严禁放到循环体内

4. 过量的 Session 问题
   这个问题，上 N+1 问题的分享中已经提及,复杂责 N+1 问题往往伴随过量 Session 问题。也就是 One More Session Per Request。
   这是非常糟糕的实践。为什么呢，为什么过量 Session 产生不好
   1. 每个 Session 都有自己的数据库连接，彼此无干扰，使用多个 Session 意味着在一个 Request 中，使用了多个数据库连接，最要命的是，这些数据库连接实际上连接的是同一个数据库服务器上的同一个数据库，实际上，只需要一个数据库连接就能完成我们所有的工作，数据库连接是非常宝贵的资源，一个能搞定的为什么要用多个，

而且还是宝贵的资源。Ok，就算用着多个连接没有问题。那么，站在数据库的角度，我能允许的最大连接数假设是 100，你一次请求就要消耗 10 个连接。那不是意味着，我同时最大只能应付 10 个你这样的用户请求。那还谈什么并发，几下就把 DB 搞死了。华通项目中确实存在 One More Session per Request 的问题，但是并不存在一个请求同时占用了多个 DB 连接的情况，因为使用了 using 块，使用完立即释放。其实，即使这么用，它还是有问题(多个 Session 产生和销毁是耗时的并且是不必要的)

2. 通常情况下，单个 Session 跟踪所有实体的变化。如果一次 request 请求中同时产生了多个 Session，那么，你怎么能保证在 Session 进行 Submmit 的过程中，哪些实体的数据同步状态的正确的。不可预料无法控制。这是一点。

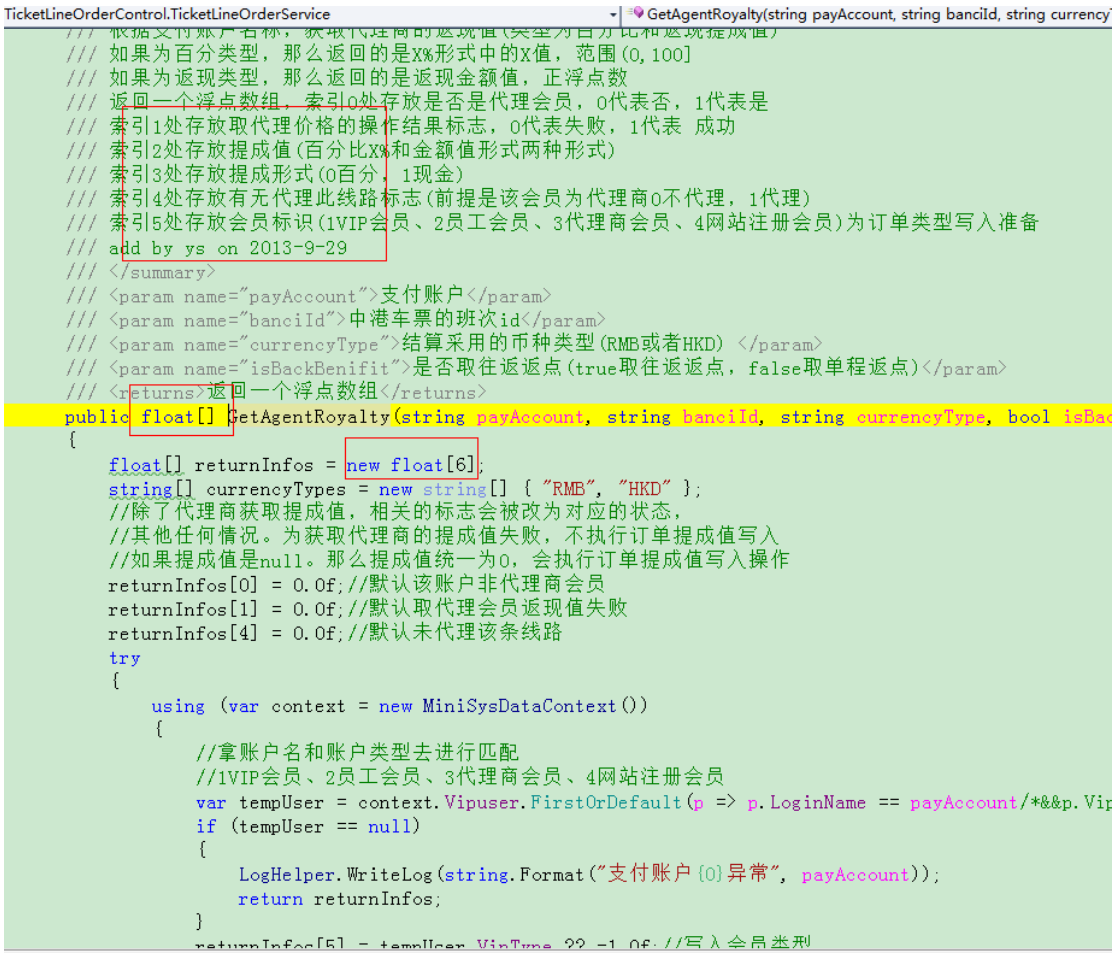
   第二点，每个 Session 只能跟踪自己的实体变化状态，它意识不到别的 Session 跟踪了哪些实体。NH 中的一级 Cache 就是 Session 中的 Cache，即，同一个 Session 内，所有实体均是透明 。不同 Session 中，实体不透明不可见。会产生什么问题，Session1 跟踪了所有 Name=ys 的所有信息，缓存到了 Cache 中，Session2 跟踪了 name=yj 的信息也缓存下来，如果 Session1 这个时候也需要查询 yj 的信息，那么，将会产生数据库查询语句，而实际的情况是，yj 信息已经被缓存下来了。只不过 Session1 看不见而已。最后一句话说明，多 Session 产生会带来内存消耗，时间消耗，并且会给 DbServer 带来不必要的查询
3. 在第二点的基础上，多个 Session 产生，NH 的的一级 Cache 很大程度上失去了作用。明明缓存下来的实体，其它 Session 看不到。

在讲完三个过量问题以后。Nhibernate 使用过程中涉及最主要的性能问题，基本上都已经介绍完了。有的性能问题并不是孤立存在的，而是彼此依附产生。就向 N+1 问题，伴随着 Session 过量问题。伴随着 Db Query 过量问题，伴随着事务使用问题。

第四篇为止，基本上讲解了一些问题是什么，为什么，怎么做。

正因为存在以上四篇谈到的问题。所以，我在 code 的时候，尽量的会去权衡合理性和性能，哪怕这可能只是皮毛上的。也哪怕这可能是错的。但是不犯错，我怎么能知道该怎么做是对的呢。没有人知道该怎么做是对的，哪来的最佳实践呢。

有兄弟跟我反映，单个方法揉合的东西太多，导致阅读性差，维护性差。就像下面的这样



在一个方法里，这个方法里我只有一次数据库访问，在这一次访问中，我拿到我要使用的或者即将使用的所有信息，通过数组返回。

这里可以看到有 6 个值，难道我应该做不止一个方法，通过多次数据库访问获取这些信息，然后再使用。显然，我觉得这是不合适的。即使我这么做可能代码看起来很丑陋，看起来不好看，不好理解，方法职责没有分离等等，有很多业界通用的法则没有遵循。但是到目前为止，我认为我的做法是正确的，也是合理的。

嗯，这些东西我接触也是尚是首次，我也希望能和其它人沟通你们心目中的最佳做法，更期待有老鸟出来指点，给出最佳实践。

当然，请回答我这三个问题，是什么，怎么做，为什么。

之前所有的分享均理论和文字偏多。所以，后期我打算拿一些我在 HT 项目发现的问题，结合 Nhibernate profile，来说明一些，生产环境中，这些问题存在性

NH 性能系列我打算再用一篇来收尾，写其它的一些零散的性能问题和使用注意要点。

有不正确的欢迎指正，也欢迎各种吐槽。有不明白的也欢迎私下来找我:)

这真是一个美好的周末。

2013-11-23