上周六参加了一场由淘宝的架构师，曾宪杰先生主讲的淘宝网架构分享。然后与主就由是了，一直没来得及总结，今晚比较有空，把这次听到的比较有启发的观点记录一下

一、为什么stateless比较有利于实现水平伸缩

关于什么是stateless的扫盲，见这个贴：http://kyfxbl.iteye.com/blog/1831869

一般有一个共识，就是把应用做成无状态的，会比较容易实现水平伸缩。但是以前一直有一个想法，就算应用是有状态的，也可以做成水平伸缩：只需要在负载均衡那里做一个session绑定就可以了，根据某种标识，把请求固定地发送到特定的server上

但是相对于有状态，stateless是更好的，主要有3个原因：

1、负载均衡不需要做session绑定，也就可以用更简单的算法，比如随机、取模等，把请求分配到任意server上，因此相对于session绑定的做法，性能会比较好

2、也是性能的问题，在7层网络模型中，session位于第7层，负载均衡如果是基于session的算法来决定要怎么分发请求，性能的损耗也会比较大

3、如果某一台server down掉了，后续的请求就没法继续往这台server发了，影响可用性

二、为什么淘宝开发session框架

如果将请求所需的信息，都放在cookie里，确实就不需要session框架了，而且也比较容易实现stateless。但是cookie也有自己的问题，cookie的大小是有限制的，还会增大网络流量（对于淘宝这种规模的应用，绝对不是一个小数），并且数据放在客户端，多少有点不安全

所以session还是要用的，但是如果session都放在app server里，stateless就不容易实现了，而且为了HA，还涉及到session迁移的问题，会比较复杂。所以淘宝自己搞了一个session框架出来

具体的实现就不清楚，不过知道了这个思路，也不是很困难。我猜应该类似于把session集中放在session server里，其他的app server都来共享就可以了。然后将session server做成集群，避免单点故障。同时session迁移的事情，也就转化成了集群的迁移

三、关于TFS

在《淘宝技术这10年》这本书里，ñ淘宝大学ò的校长神话了这个技术。那天据曾宪杰先生说，TFS只是比较擅长处理小而多的零散文件，单机部署时性能也不是很好，但是在集群部署时，性能和可用性，确实有一点优势

这块我没有怎么研究过，当天曾先生也没有深入讲，不是很了解，就不多说了

四、淘宝子应用之间的集成，为什么不使用ESB

淘宝在11年左右开始拆分应用，走ñ服务化ò方向之后，应用拆分了很多个独立的子应用（大约有1000个左右）

这么多的应用要在一起完成业务，势必涉及到集成的问题。但是淘宝没有使用ESB，而是用了自研的服务框架和消息中间件

这主要是因为ESB更擅长处理异构系统的集成，而淘宝这将近1000个应用，基本都是JAVA应用，所以ESB的这个优势不明显；另外，增加机器水平伸缩，是淘宝的杀手锏，如果采用ESB架构的话，那在水平伸缩的时候，就连ESB那层也要一起伸缩，比较麻烦

ESB的这个优势不明显；另外，增加机器水平伸缩，是淘宝的杀手锏，如果采用ESB架构的话，那在水平伸缩的时候，就连ESB那层也要一起伸缩，比较麻烦

五、特性开关和优雅降级

稍微提到了一点，即在应用中放一些开关，在流量告警的时候，关闭一些功能，以主动避让的方式，避免系统压力过大而不可用。我理解类似于丢卒保车，在万不得已的时候，牺牲一些次要的业务，保护主要的业务

具体的实现没有提到，目前好像我们也没碰到过这种场景，先简单记录一下

六、分布式的好处

我原本认为，在满足条件的情况下，分布式架构会提升性能（条件是CPU是性能的瓶颈，以及大的任务可以分割成可并行的多个子任务）

不过那天曾宪杰先生反对这个看法，他认为分布式只会把本地调用变成RPC，带来额外的传输损耗，所以不但不会提升性能，反而在大多数场景下，会降低性能

我觉得不尽然，有时候分布式应该还是能提升性能的，对于那种CPU密集型的计算来说。比如NASA不是有一个项目，是利用很多个人PC，来计算小行星数量还是什么的。这个好像又叫啥网格计算。不过，我们做的大多都是企业应用，或者互联网应用，因此基本上CPU不是瓶颈（瓶颈主要在IO）；由于业务逻辑的关系，也很难说就能拆分成可并行的子任务，所以场景不太满足。因此对于曾先生的这个看法，我还是相当认可的

那么，既然分布式对于性能有减无增，为什么还要用分布式呢，总结了下面3个原因：

1、组件复用。这个很好理解，就不用多说了

2、提升开发效率。淘宝把一个应用拆分成很多子应用以后，就可以实现ñ小团队维护小应用ò，做到了ñ专人专事ò，效率比较高。此外，小应用显而易见，也更容易维护

3、实现ñ差异化水平伸缩ò。这个词是我自己瞎造的。考虑这种场景，数据库只能支持10个连接，但是需要20个系统才能支撑http并发。那么这时候应该部署几套应用来组成集群呢？10个肯定不行，并发撑不住；但是20个也不行，因为每个应用都直接连数据库，又超过了数据库连接的上限

如果实现了分布式，那么就可以这样：部署10个service server，来连接数据库，对上层提供服务；部署20个app server，来响应http请求，不直接连接数据库。挺巧妙的，架构上也挺有美感

当然，数据库的连接数上限肯定不只是10个，上面只是打个比方。从侧面也看出，淘宝的业务量大到了一个超乎我想象的程度ð ð 居然连数据库的连接都不够用了

同时，这么多子系统要调来调去，是很复杂的（1000个子系统，想想都觉得很复杂）。所以淘宝开发了服务框架和消息中间件，来解决这个问题。或者说，正是先行一步有了这些基础框架，服务化拆分的路才能走得比较顺畅

另一方面，分布式架构也是有坏处的。比如部署和调试都变得复杂了，淘宝有专门的运维工具和开发工具，来减少这种痛苦。提到了一个google的Dapper，和淘宝自研的eagle eye。有需要的时候可以研究下

七、数据库的水平伸缩

相对于app的水平伸缩，数据库伸缩是比较复杂的。因为RDBMS本质上是单机系统。虽然可以部署N台

也说数据库的水平伸缩

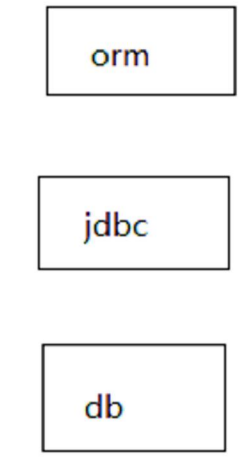
相对于app的水平伸缩，数据库伸缩是比较复杂的。因为RDBMS本质上是单机系统。虽然可以部署N台db server组成集群，但是主要是保证了HA，也支撑更多的连接数。但是如果单个数据库里的数据量太大的话，读写都会变得很慢，还是瓶颈

最后没有别的选择，只能拆表，比如把user拆分成user1和user2（拆分的维度可以很多，基本上是水平拆分）。这就带来2个问题，第一是编程变得复杂了，简单的ORM+jdbc就搞不定了；第二就需要开始处理数据迁移的问题

针对这2个问题，淘宝的答案是，用TDDL做数据库路由，对上层应用保持透明；开发专门的迁移工具，在拆表后做数据迁移（尽可能缩短业务中断的时间）

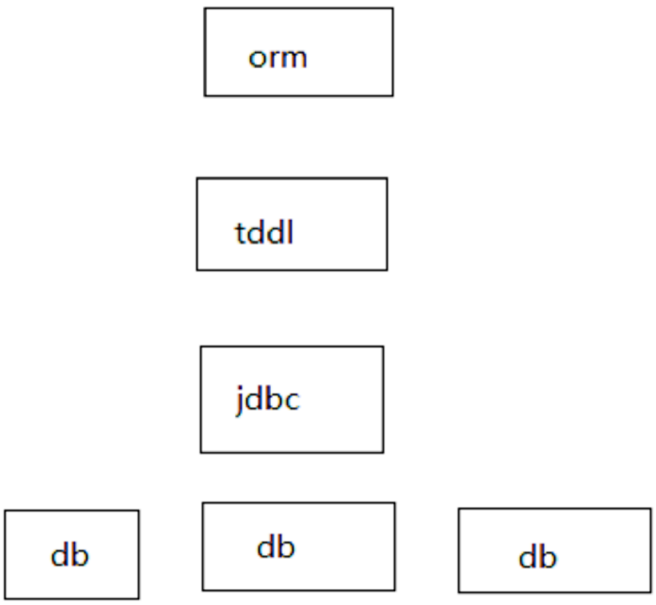
TDDL没有开源，类似的有一个叫cobar的框架

大体的思路应该是这样的。本来应用的DAL层，直接就建立在JDBC之上，JDBC去连具体的数据库



但是这样的话，应用就依赖底层的数据库。当需要查询一条数据的时候，应用需要知道应该去哪个表（或者哪个数据库）里查询；在数据库伸缩的时候，应用的代码也要跟着改才行。这样肯定是不大好的

所以，在DAL和JDBC之间，又增加了一层，也就是TDDL



现在，DAL不再是直接依赖JDBC，而是依赖TDDL。数据库路由的功能，都在TDDL里完成，底层数据库伸缩的时候，也是在TDDL里进行ñ配置ò，应用的代码不需要变化。果然是应了那句话，ñ计算机的问题，大多可以通过增加一个抽象层来解决ò

TDDL的实现不清楚，也没有开源。不过我猜测应该也是封装成一个DataSource，适配JDBC规范，这样其上的ORM框架，也就可以直接使用了。然后在TDDL内部，会去解析SQL，处理配置文件（数据库路由），并且依赖厂商JDBC的实现，去真正连接底层的数据库

总的来说，淘宝这个基于TDDL的方案，还是在解决RDBMS水平伸缩的问题。应该也正在引进NOSQL的方案，互为补充

八、关于虚拟化

淘宝内部也用到了虚拟机，也就是私有云。大致有几个好处：

1、硬件一虚多，提升硬件利用率，降成本

2、应用和具体硬件隔离，便于维护

3、硬件的内存太大时，JVM对内存的管理不是很好，不如把一台硬件虚拟化成多个虚拟机

淘宝非常重视对硬件和应用状况的监控，好像由于历史遗留原因，监控的工具很多是针对单个进程的。所以目前常见的做法，是一台虚拟机上只启一个JVM（单进程），和过去的监控工具相兼容（这里没有详细说，刚好那会我也有点累了，听得不是很仔细，不知道有没有理解错）

另外淘宝用的虚拟化工具是linux container，好像跟VMware那套解决方案也不太一样，不是完全的虚拟机，而是更轻量级的硬件资源的隔离，后面可以再研究一下

九、OSGi在淘宝内部的使用

现在基本不怎么用了，OSGi主要的价值，在实际中体现得不太明显

比如类隔离，用更简单的自定义ClassLoader也可以实现；单机多版本服务，用的场景也很少；热部署也不是很实用

但是，基于OSGi框架做开发，复杂度的上升又是显而易见的。因此，用很高的代价，只能换来较少的收益，在开发人员之间推动很困难，渐渐地就不怎么用了

我们之前的一个产品，也是类似的情况。公司内部一个平台，三年前的一个主要卖点就是OSGi架构，好处也就是OSGi官方宣传的那一套，而现在最新的版本也在ñ去OSGi化ò，有一种走了弯路的感觉

我个人觉得，除了明显增加开发复杂度之外，OSGi还有一个问题，就是和java ee规范的兼容性，离完美还是相去甚远。就连最简单的一个问题，ñOSGi框架与servlet框架嵌套ò，现在虽然有方案，但是同样相当复杂。我觉得对于OSGi，目前还是保持适度的关注就可以了

十、去IOE化，后续的趋势

这一节本来不想写，是关于淘宝去IOE化，以及后续的技术规划。感觉没有什么新东西，但是大领导就喜欢听这些，还是先记一下，后面写脚片说不定有用

这一节本来不想写，是关于淘宝去IOE化，以及后续的技术规划。感觉没有什么新东西，但是大领导就喜欢听这些。还是先记一下，后面写胶片说不定有用

淘宝有一个动作是去IOE（IBM的小型机、Oracle数据库、EMC高端存储）。早期的淘宝，主要靠的是高端硬件，思路就是用钱解决问题。但是这样做的成本很高，淘宝的业务规模又扩张得太快，所以要用更经济的方案。用PC Server、MySQL数据库等，通过大量廉价的硬件，水平伸缩组成集群，来替代昂贵的硬件，思路就是用技术解决问题。跟google的思路一致

淘宝V4.0的几个技术趋势：页面组件化、私有云部署、多终端。没有什么新东西，实际上我们2012年也在考虑这几个规划，那天听到曾宪杰先生说淘宝也想做这么几件事，我觉得有点惊讶，怎么这么巧。实际上我们产品目前的规模（并发和数据量），连淘宝的零头都不到，因此也不是很急迫，应该是预研性质的

十一、总结

本文大致记录了那天的收获，有些内容很受启发；另外一些当前不太用得上，先记录下来留着以后再想想

总的来说，我对曾宪杰先生的水平很佩服，淘宝架构师确实是名不虚传

此外还有一些技术之外的感悟，主要有2点

一整天的交流下来，我能感受到淘宝一种很务实的精神，我感觉这是我司目前所缺少的。曾先生多次提到ñ先work，再优化ò、ñ当时也没想那么多，只是先解决问题，不然就死了ò、ñ专人专事，小团队负责小系统ò，我觉得很朴实，但是很有道理

反观我司，现在动辄就100多号人做一个小产品，想用数量弥补质量的不足。实际上我觉得反而是降低了效率（这些产品我觉得10个人左右的小团队，完全可以做得更好），同时也造就了大量的领导岗位，催生了各种流程，进一步降低了效率

还很喜欢搞ñ架构ò，花了很多的资源，最终却没有在产品中落地，带来什么业务价值，只是把产品做得更烂，同时制造了一些ñ专家ò

我觉得淘宝也有这个趋势。公司慢慢大了，优越感就来了，夸夸其谈的风气慢慢也滋生了。只能说，并不是话多声音大的就是专家，各人有各人的活法。个人感觉，淘宝慢慢也会走向我司目前的状态，可能这是大公司的通病，也是人的通病。。

另外一个感想，就是觉得淘宝现在有这么一批技术高手，很大程度上是业务造就的。所谓时势造英雄，业务规模不断扩大，逼迫技术必须进步，解决业务问题，反过来也促进了业务的发展

而作为早期的淘宝技术人员，见招拆招，解决各种技术问题，慢慢也就成为了高手。所以，应该要承认淘宝技术的强大，但是也要客观看待，没必要盲目神话和崇拜。淘宝做的也就是一个规模巨大的商城，不是原子弹或人类基因图谱什么的

再次感谢曾宪杰先生，水平很棒，表达得也很好。下次有类似的机会，还会去参加