【技术分享】刘世勇：链家MySQL高可用架构设计

原创 2016-08-23 ZYY 数据库技术大会

本文整理自DTCC2016主题演讲内容，录音整理及文字编辑IT168@ZYY，@老鱼。如需转载，请先联系本公众号获取授权！

演讲嘉宾

刘世勇

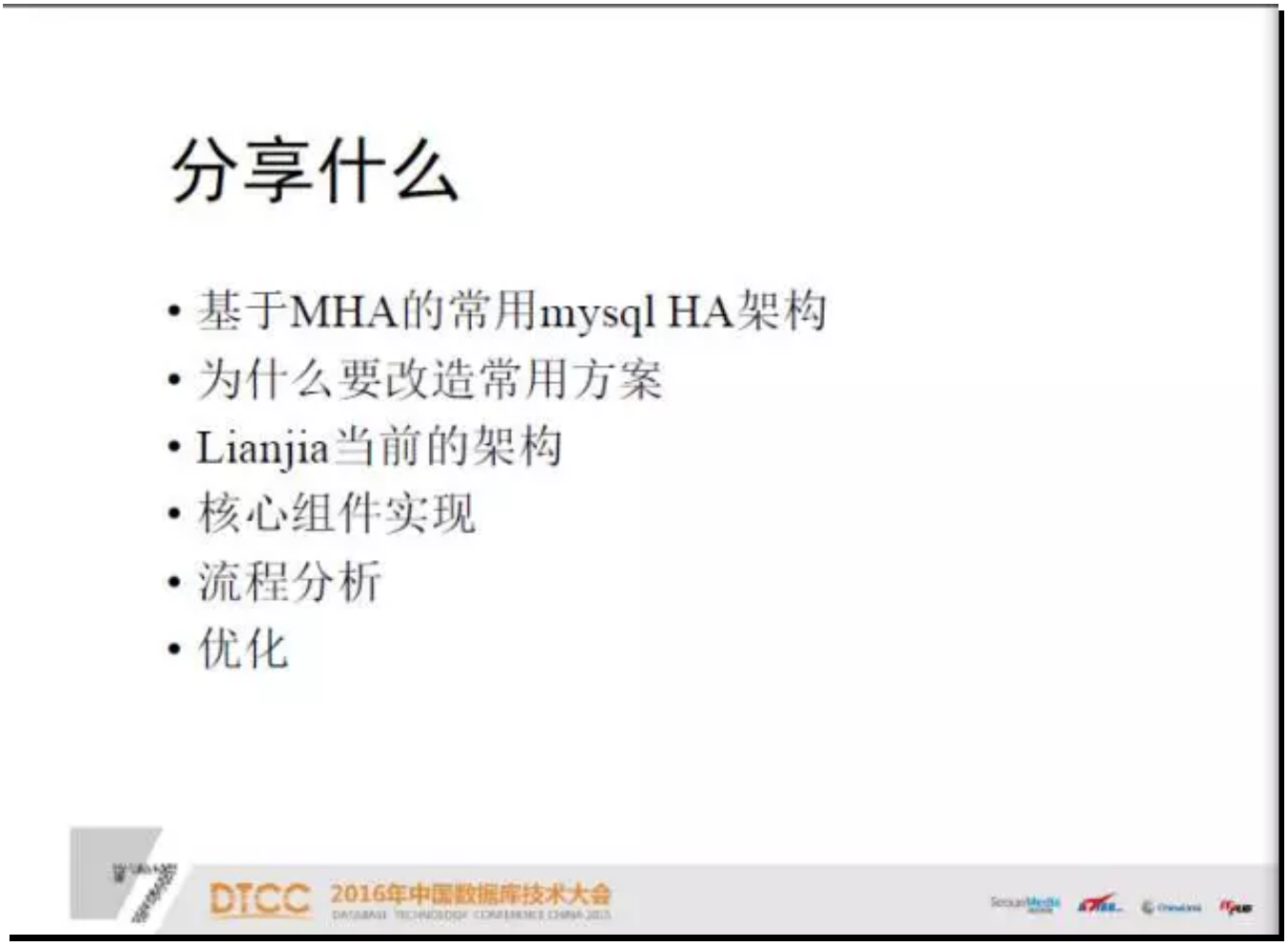
链家网DBA

2011年毕业于四川大学，先后混迹于华为、网易，2015年1月加入链家网，目前主要负责链家网oracle和mysql数据库的运维，包括数据库架构设计，DB性能调优和SQL优化，DB自动化运维平台的构建等工作。

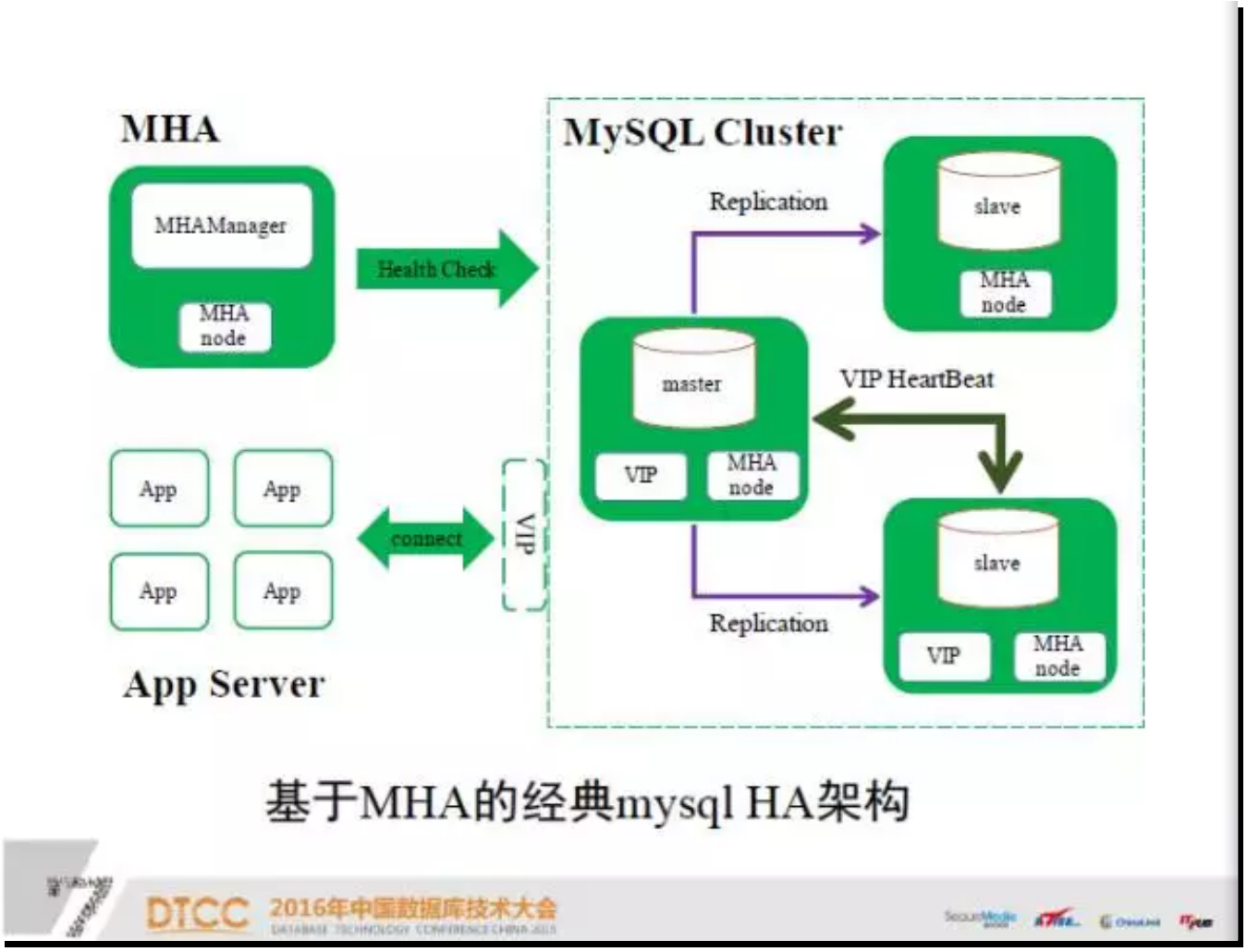
分享内容

大家下午好！非常荣幸能够在这里跟大家分享，我是来自链家网的刘世勇，我今天分享的主题是《基于Zookeeper+MHA的mysql高可用架构设计》，其实今天主要跟大家提一下思路，希望能够给大家一些启发，今天不会特别讲架构，我们只是设计了一个思想模式。

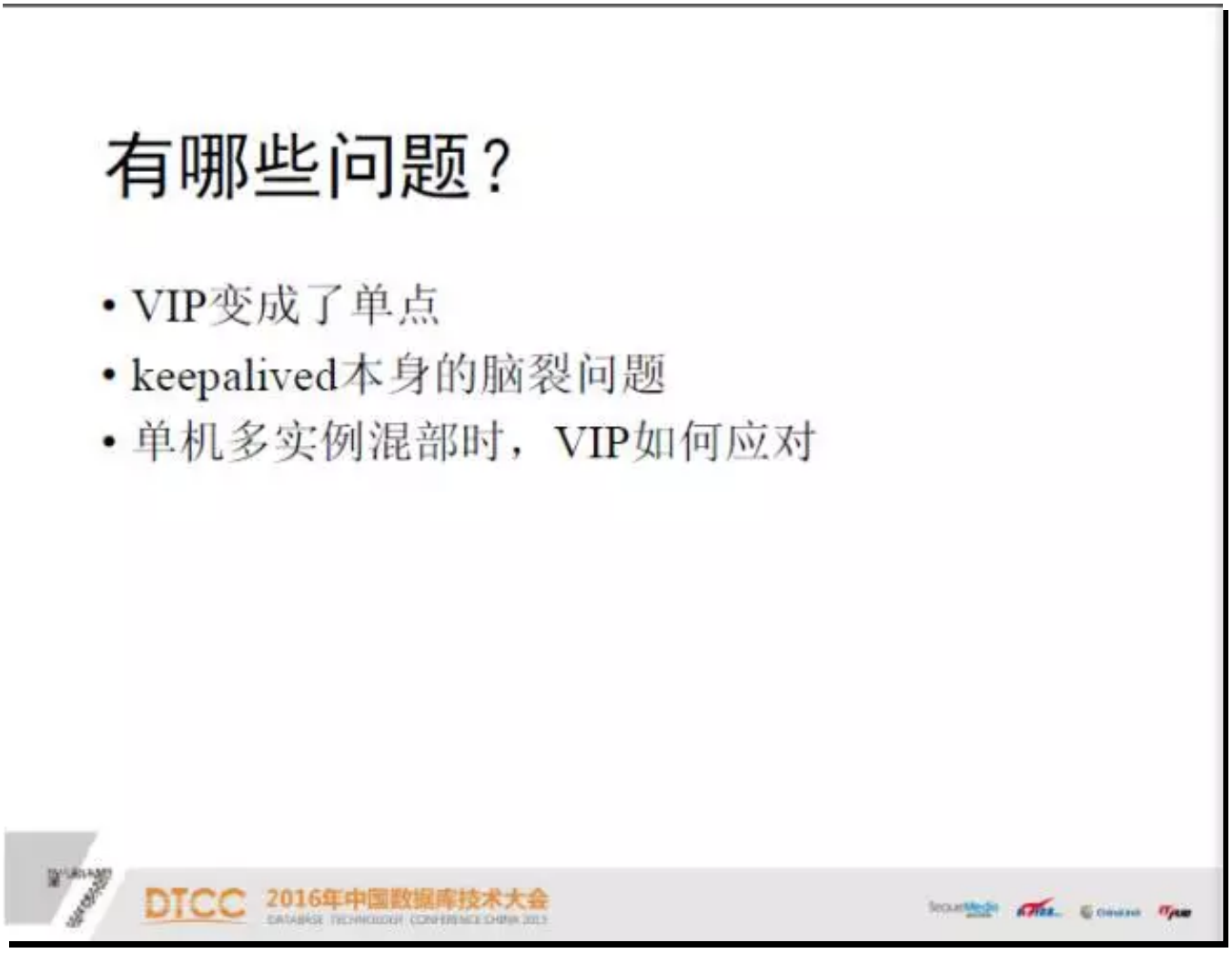
接下来50分钟分享什么呢？主要分为六个方面，第一，基于MHA的常用mysql HA架构；第二，通过分析常用架构的一些缺陷，引出我们做架构改造的原因；第三，简单放一个架构图；第四，深入剖析当前链家里的核心组件实现；第五，梳理整个架构流程；最后说一下优化，具体的如图所示：



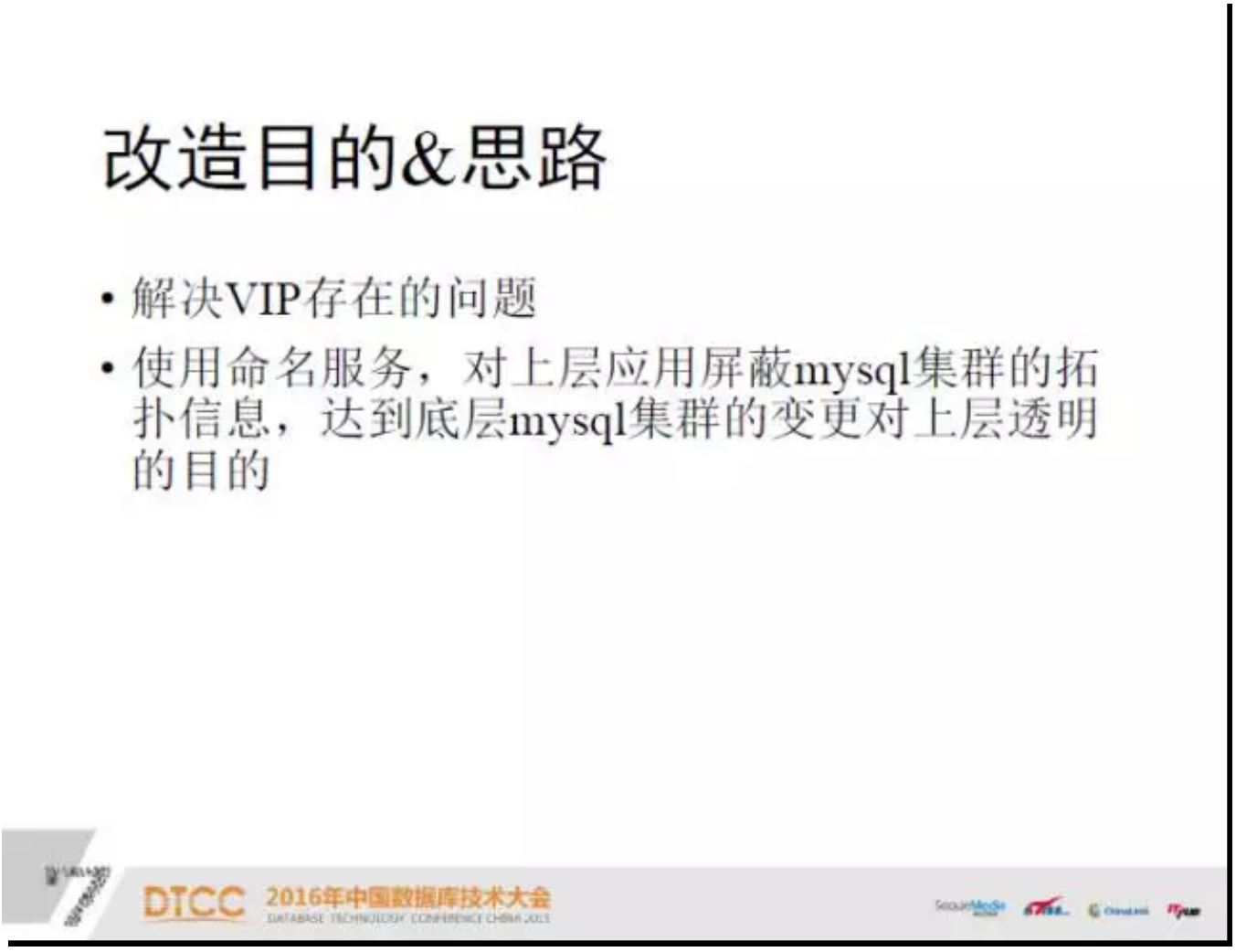
首先是基于MHA的常用mysql HA架构，我写的是经典架构，其实是比较通用的方案，通过Health Check监控Mysql集群，这有一个对外的VIP，也就是对上层应用，有可能有多个比如写一个，读一个，通过VIP连接Mysql集群。在Mysql集群内部，有主和备主，它们之间存在一个VIP HeartBeat，VIP自动挑选备主，达到高可用的目的。



这个架构存在什么问题呢？有如下三个问题，一是VIP变成了单点，比如现在VIP提供者异常挂掉，虽然此时Mysql集群是存活的，但对上层应用来说没什么区别。



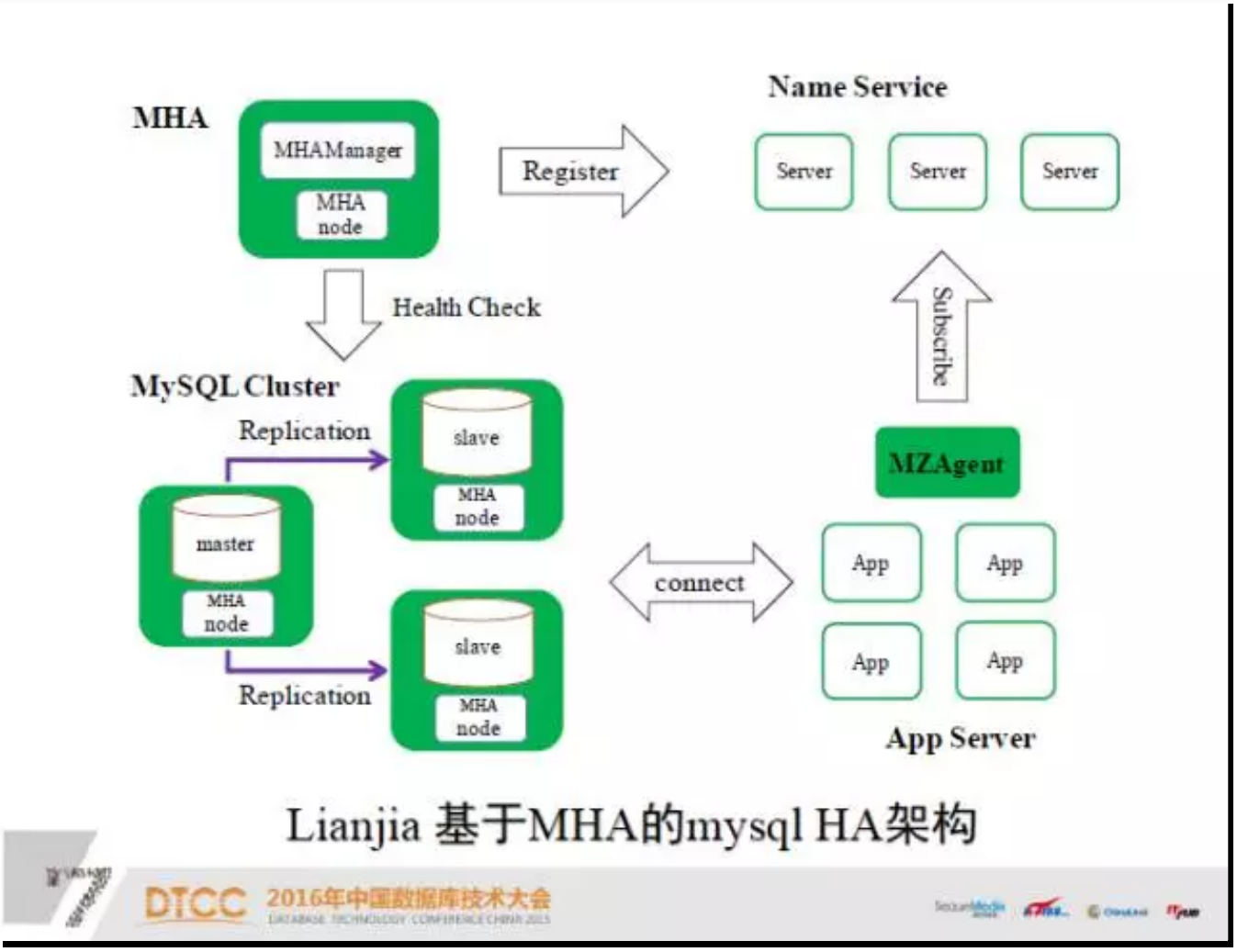
二是keepalived本身的脑裂问题，这会造成什么结果呢？第一，上层应用到数据库的连接不稳定，比如VIP一会在主，一会在从，有时候可写，有时候不可写，这会造成上层应用不稳定；第二可能造成数据脏读；第三也是我们面临的一个主要痛点，就是单机部署多实例时，VIP有两种应对方案，第一种是，可能在一个系统挂一个VIP，可能所有Mysql集群都用该VIP连接。但会出现问题，比如其中某一个集群切换了，那其他集群是不是也要跟着切？实际上对其他集群来说，这是计划外的，但因为架构有缺陷，导致不能切。二是，给每一个Mysql集群分配一个VIP或多个VIP。当一个集群发生切换时，要更改配置文件，然后重启，如果每一个集群都想切换，就需要多个配置文件，配置文件怎么管理是一个大问题。第二是做VIP时，主从之间需要协商谁持有VIP，在协调过程中，VIP对上层应用是不可用的。最后一个问题是IP浪费问题。



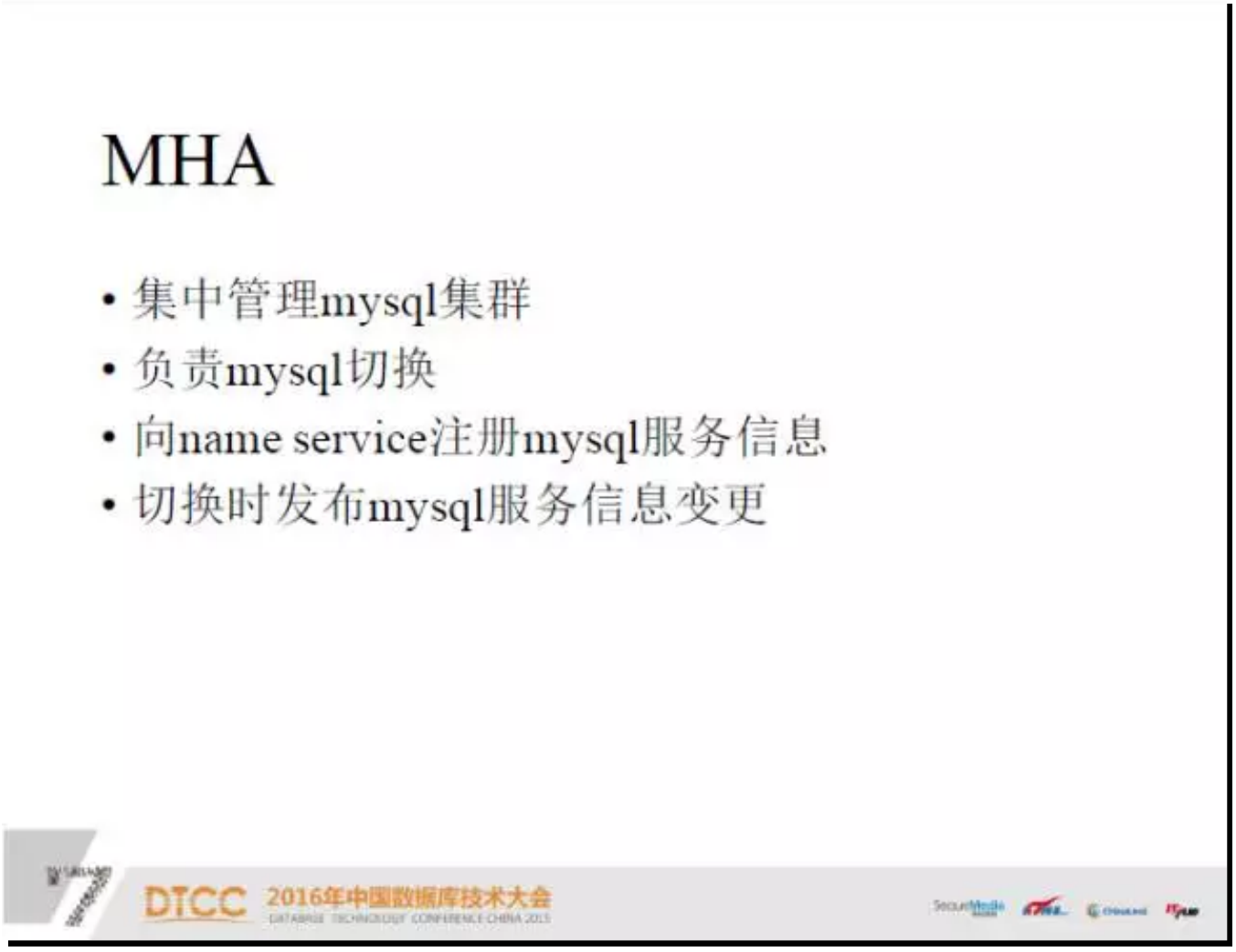
当读和写各需要一个VIP时，这个问题会尤其严重。我们肯定要想办法解决这个问题。这里，主要说一下链家的改造思路——引入命名服务。用命名服务替代VIP，对上层应用屏蔽Mysql集群的拓扑信息，达到底层Mysql集群的变更对上层透明的目的。命名服务用一句话概括，就是通过指定名字获取特定资源，把Mysql本身当成一种服务提供给上层应用。因为上层应用的一些接口可以作为服务，所以Mysql也可以作为服务。

我们改造是为了达到两个目的，一是对上层应用屏蔽Mysql集群的拓扑信息，二是使底层Mysql集群的变更对上层应用透明。

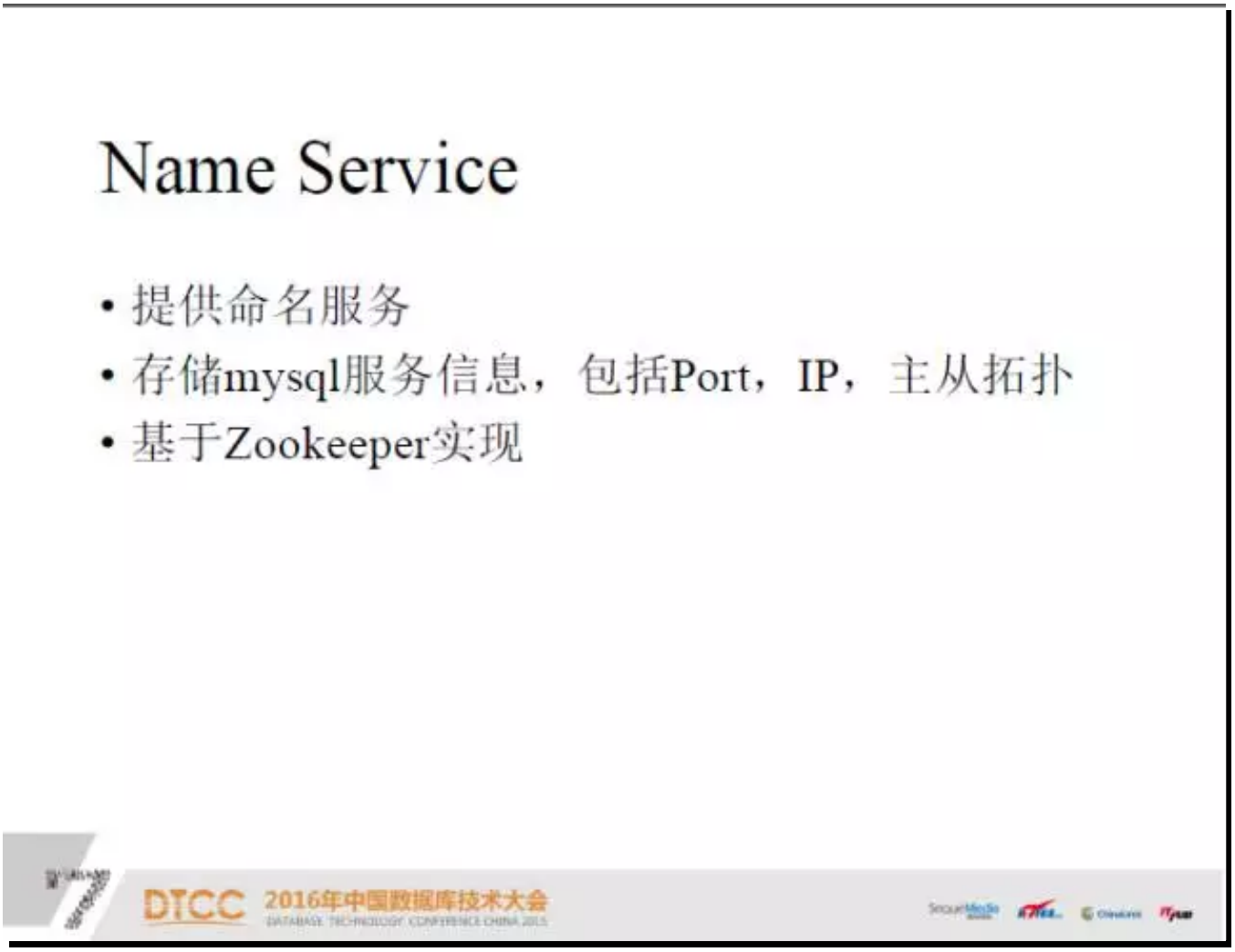
以下是链家当前的改造架构，跟通常架构的区别在于我们去掉了Mysql集群里的VIP，加入了Name Service，其实这里可能有一些企业或者叫命名服务提供者。App Server是应用服务器，对应用服务器我们起了一个MZAgent。



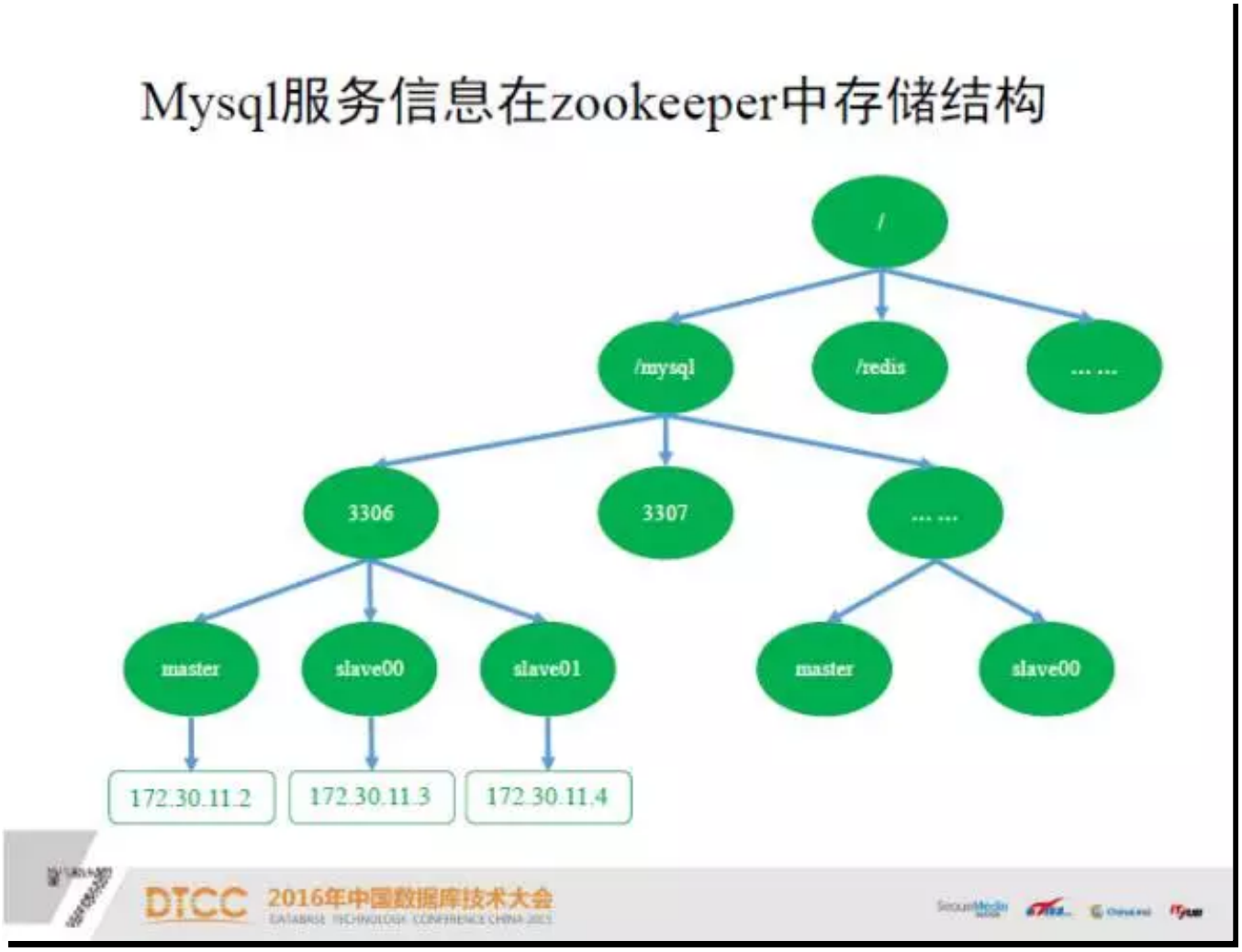
再看一下整个流程，MHA是注册，注册Mysql本身的服务信息，agent是我们自己开发的功能，它可以从命名服务提供者订阅Mysql服务信息，然后根据拿到的名字连接Mysql。



接下来看一下核心组件的功能和实现。MHA本身提供集中管理Mysql集群和负责Mysql切换的功能。我们加入了向Name Service注册Mysql服务信息的功能，通俗点说，就是当MHA对Mysql集群监控启动时，可以根据Mysql集群的配置，把Mysql集群的服务信息写到Name Service上，当然在这里就是Zookeeper。我们加入的第二个功能就是MHA做Mysql集群切换时，发布Mysql服务信息变更。之所以用发布这个词，主要是基于Zookeeper发布定位的机制和模型。通俗一点说，就是在Mysql集群做切换时，把命名服务提供者的Mysql信息更新一下。



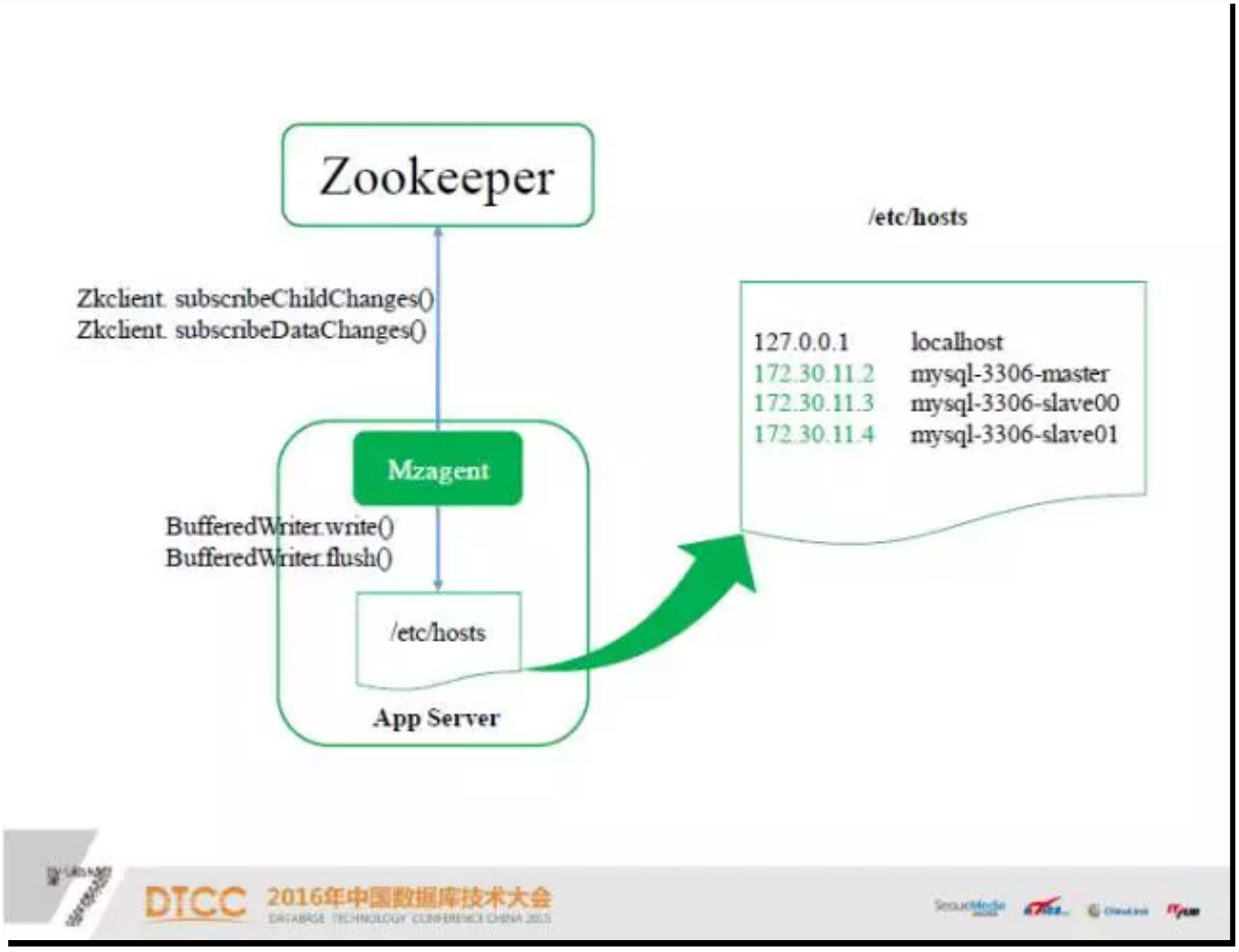
Name Service就是命名服务提供者决策，它对上层应用提供命名服务。另外也相当于是Mysql集群服务信息的中央仓库，集中存储了比如公司所有的Mysql集群的服务信息，包括端口、IP、主从拓扑关系等。



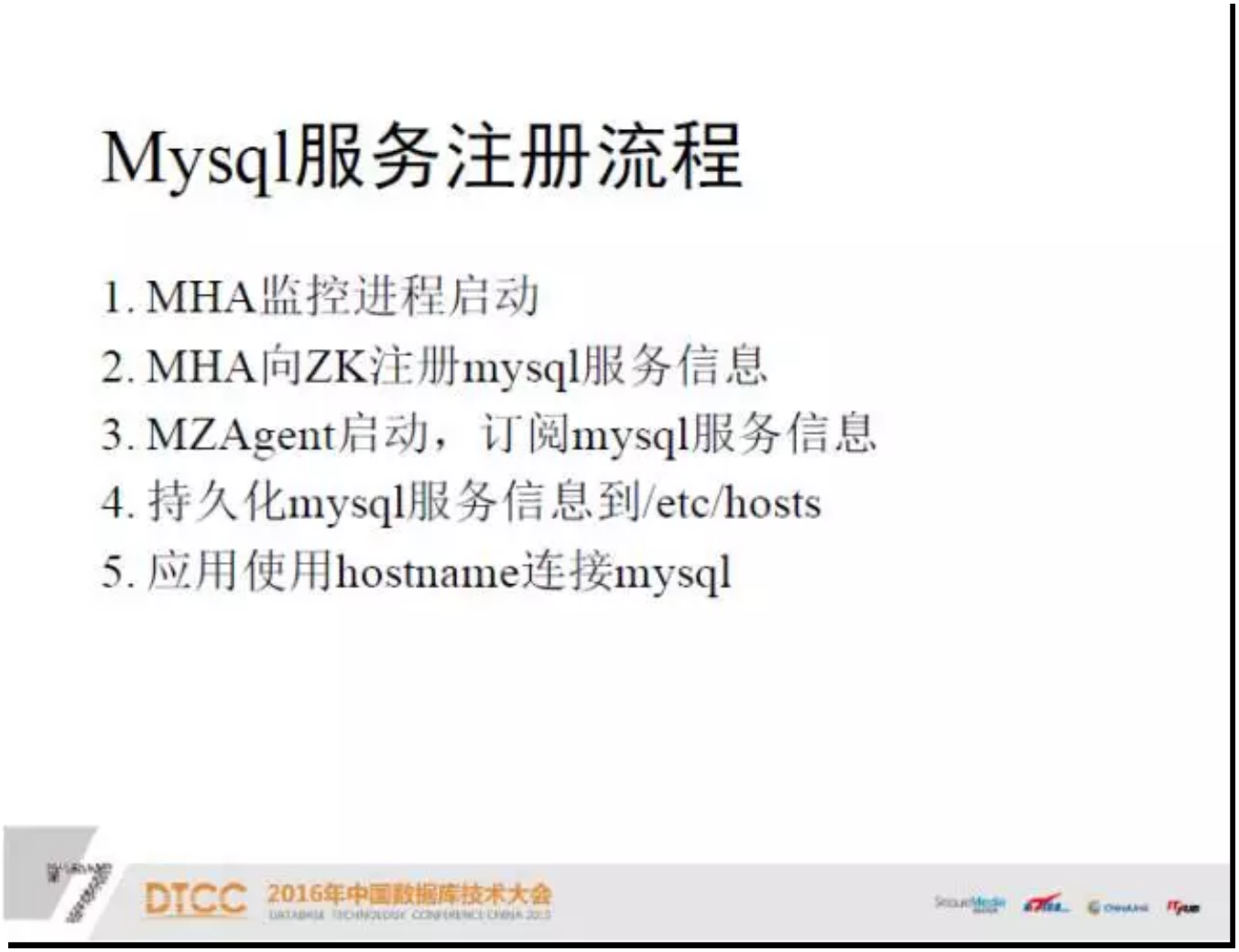
下面看一下Mysql服务信息在Zookeeper中的存储结构，是树状结构还涉及一些应用的运维服务。/mysql下面是3307，链家有一个约定俗成的规则，就是通过端口号唯一标识Mysql集群，除了端口号，也可以用其他任何你能够理解的集群名字。在这个集群下面有三个子节点，第一个子节点实际上是集群里Mysql实例的节点，最下面这个方框里的节点所对应的value是这个实例的IP。通过这种方式，可以把整个Mysql服务信息存在主机上，可以把这个路径当成名字提供给上层应用，应用能够根据这个路径查到叶子节点的value，根据名字解析到IP之后创建相应的数据库连接。

接下来是MZAgent，这是我们自己用Java写的代理，不存在app server，它为app server提供订阅、消费Mysql的服务信息。像Name Service一样是按需的，订阅之后，持久化本地/etc/hosts。我们为什么考虑用host文件，一是能够让MZAgent本身的逻辑尽可能简单，二是这样对应用是无切入的，推进Mysql高可用方案，业务不需要做任何改造，只需要把文件里面的IP换成提供的Mysql集群的名字即可。

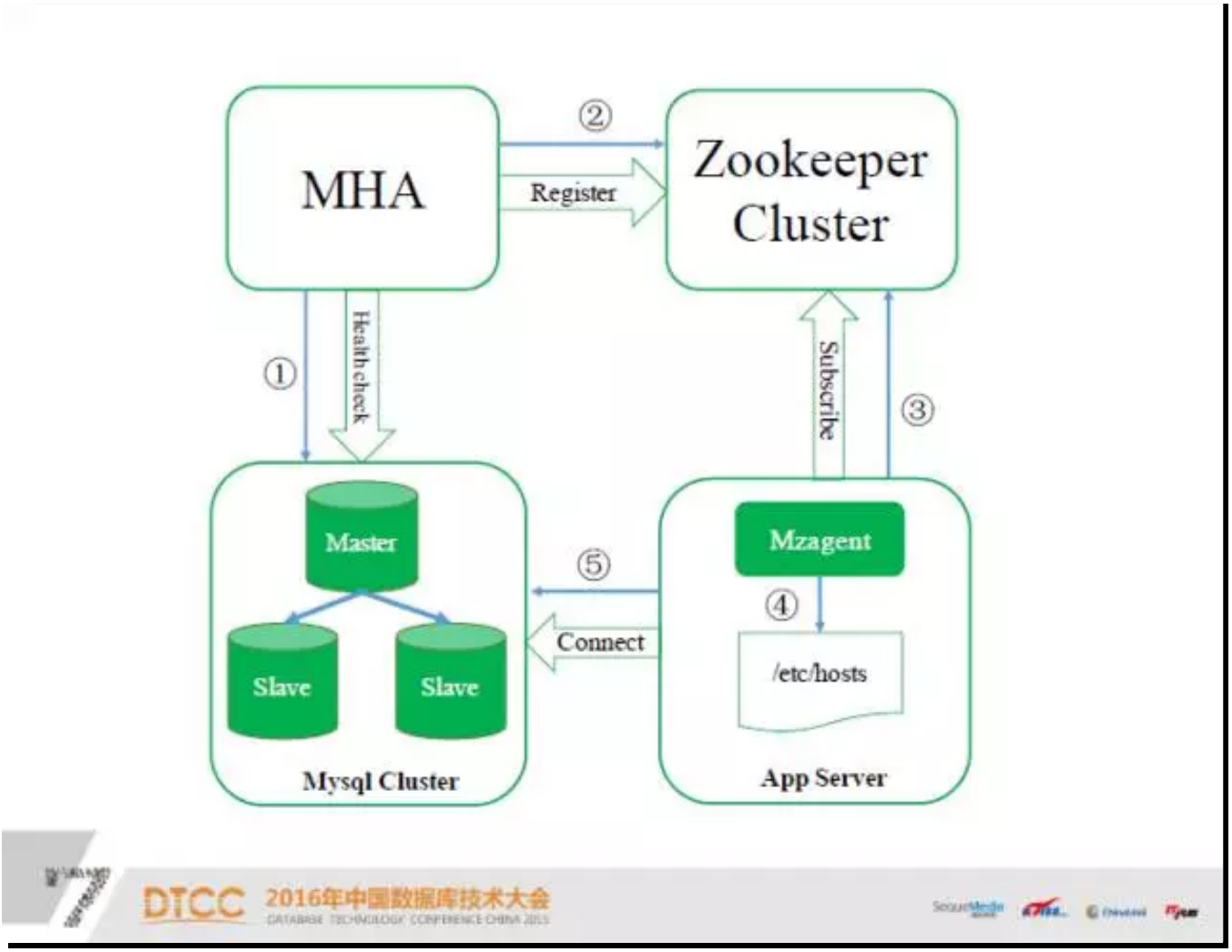
第二个功能是订阅变更，底层Mysql集群切换之后，我们把运维服务提供者上的集群信息进行了更新，MZAgent能够实时订阅到更新，同时修改本地host文件。MZAgent主要是基于zkclient来使用的。zkclient是一个开源Java的Zookeeper客户端，它在原生客户端上扩展了很多对开发者有用的功能。



ZKclient有两个函数，主要是用来写host文件，一是subscribeChildChanges( ) 函数，主要用于订阅节点变更，比如集群扩容，添加索引。二是subscribeDataChanges()函数，主要作用是订阅叶子节点的值的实例，比如给IP做切换，一旦IP变了，这个函数就能订阅到变化，同时修改对应IP。

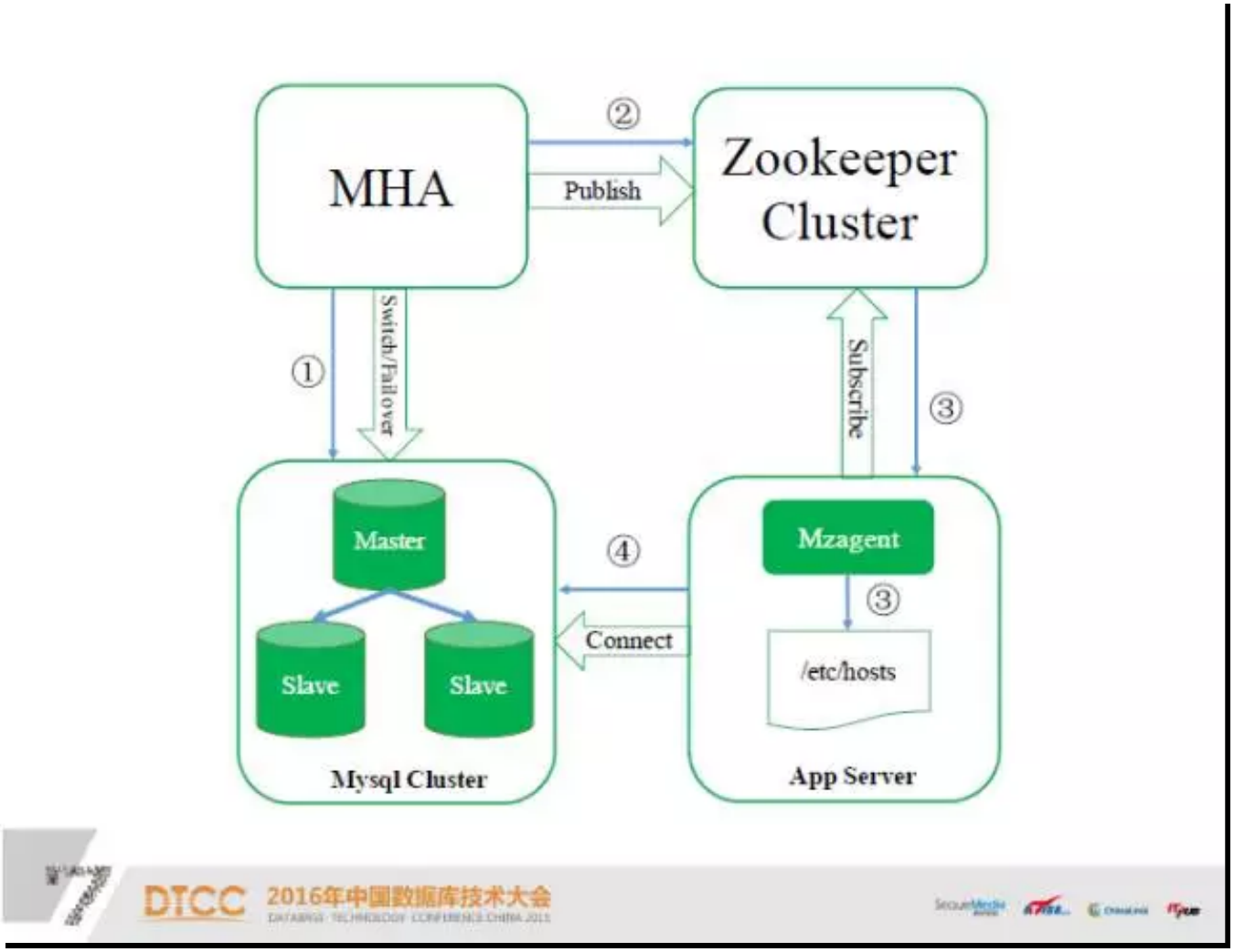


说完核心组件，我们来看一下架构流程。首先看一下Mysql服务注册流程，什么叫服务注册流程，怎样把Mysql服务信息写到命名服务上提供应用者使用，接下来都会给出答案。

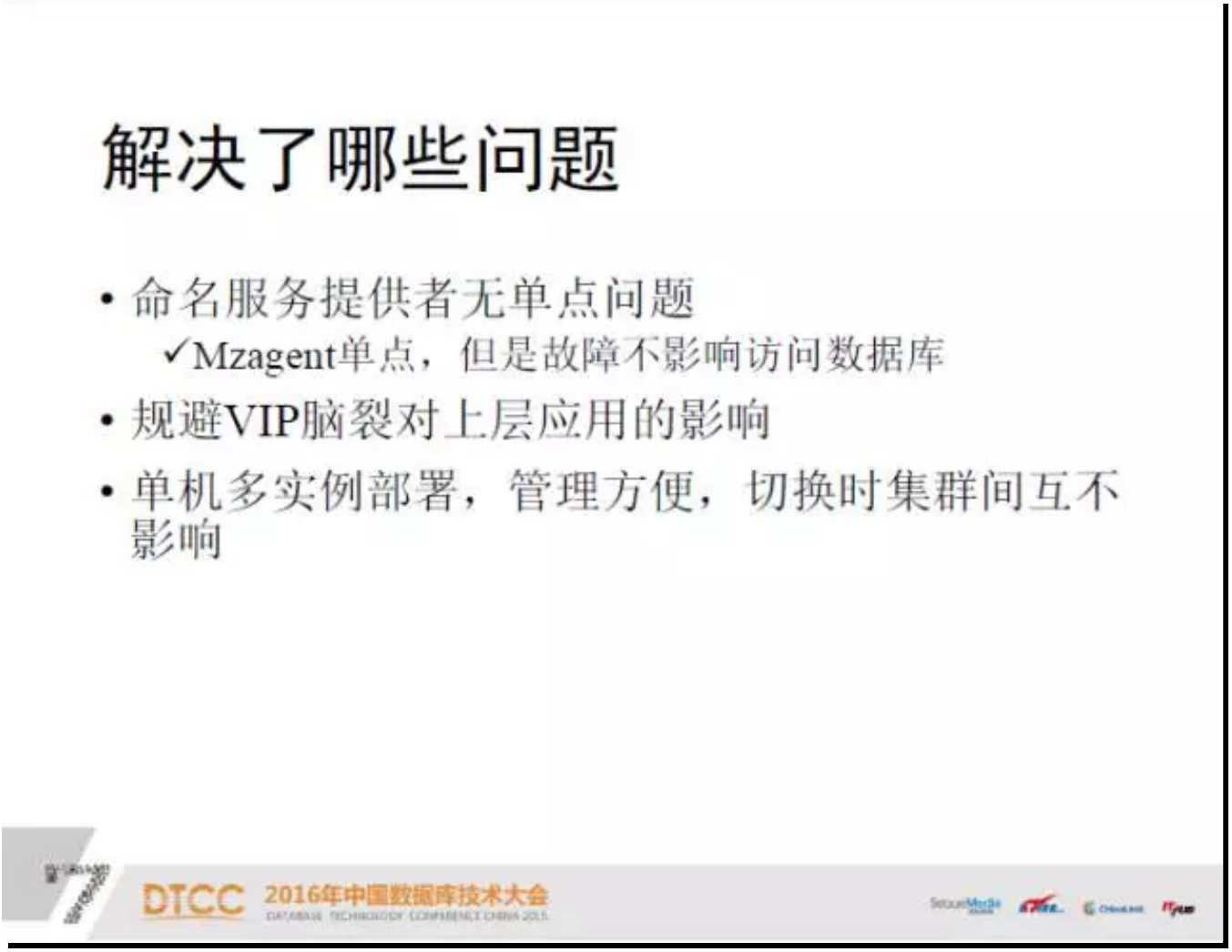


一是ZKclient监控Mysql集群。二是，监控启动后，它会向Zookeeper注册，就是说把Mysql集群的配置信息写到模块里。三是，启动应用服务器上的MZAgent进程之后，订阅Mysql服务信息。主要回答了如何对上层应用屏蔽Mysql集群的底层物理机的问题。

下面看一下Mysql的切换流程，Mysql集群切换之后，应用能够实时获取到更新之后的Mysql集群信息。接下来是整个流程图：



一是切换，二是publish发布，三是订阅变更，这时上层应用不需要做任何更新。名字还是同一个名字，但对应的IP已经变了，这时创建一个切换之后的新连接，这个流程回答了Mysql集群如何对上层应用进行切换的问题。



现在这个方案解决了如下问题，第一，命名服务提供者无单点问题，首先MZAgent部署在客户端，应用服务器上，这个应用对它没有产生依赖只是挂着，但如果MZAgent宕机，不做任何Mysql切换是不会影响上层应用的。第二，规避了VIP脑裂对上层应用的影响，第三，解决了我们的主要痛点，单机多实例时，管理起来非常方便，只要简单的在Zookeeper上注册，集群切换也不会相互影响，解决了IP资源问题。

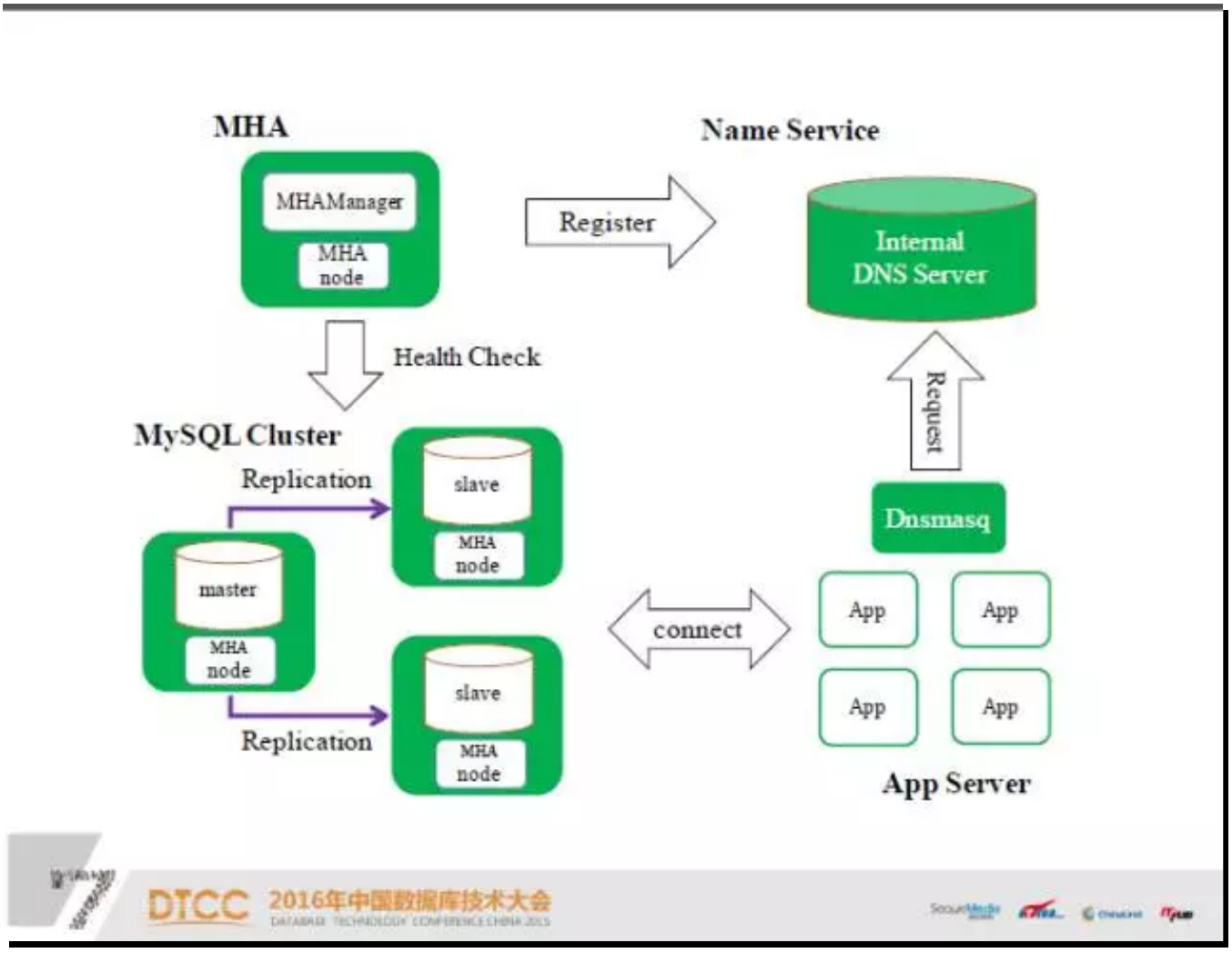
说了这么多，这个方案是不是就真的很好用，在实际运维过程中，就没有任何问题了吗？肯定不是！我们下面看一下在运维这个架构时发现的一些问题。

一是Mysql集群扩容或缩容时，应用需要做相应地配置更新，比如应用集群Mysql集群加了一个词，我想要把它从上面择下来，这时肯定要去hostname里操作，但如果使用了hostname，应用肯定要做配置文件更新，如果添加了一个节点，就要把新节点的hostname添加进去，才能够使用。二是/etc/hosts容易误操作，可能导致应用访问DB异常。如果是一些线上操作，如果把hosts删掉，数据库迟早会出问题，因为无法解析hostname。

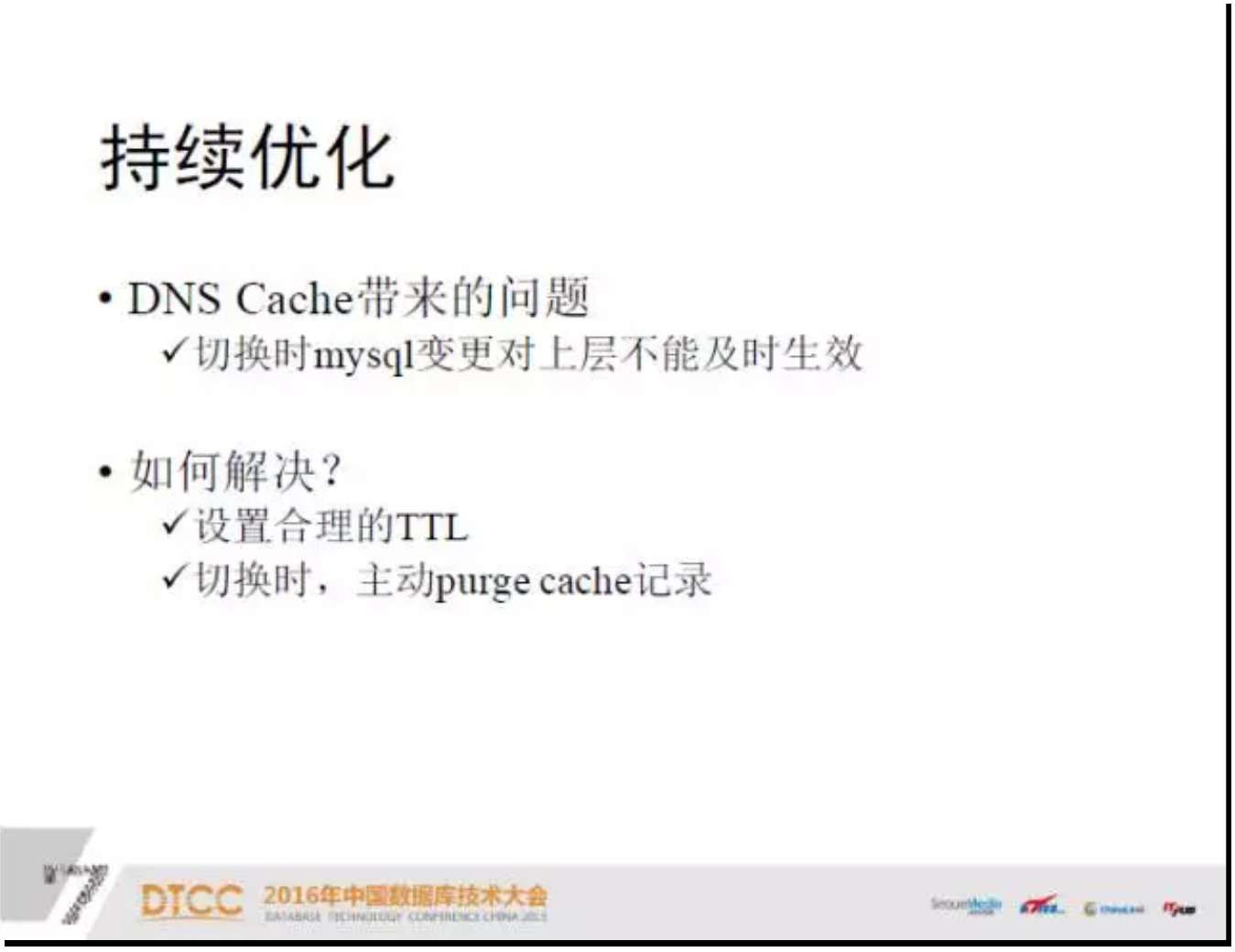
三是一些强加的问题，比如APP Server是按需订阅的，这时会出现一个问题，就是不同业务线的APP Server做一些个性化配置。比如对应用扩容时，什么都做好了，突然访问数据库异常，这时才想起来还有一个agent配置。当然agent需要自己去维护，代码新版本发布之后，这些都是额外的工作量。

针对上述问题，目前的解决方式是使用DNS，就是把Mysql服务名字解析拿到DNS上，DNS做这些事有一些自己的先天优势。直接用内部DNS提供命名服务。二是为每一个Mysql分配一个内部域名。

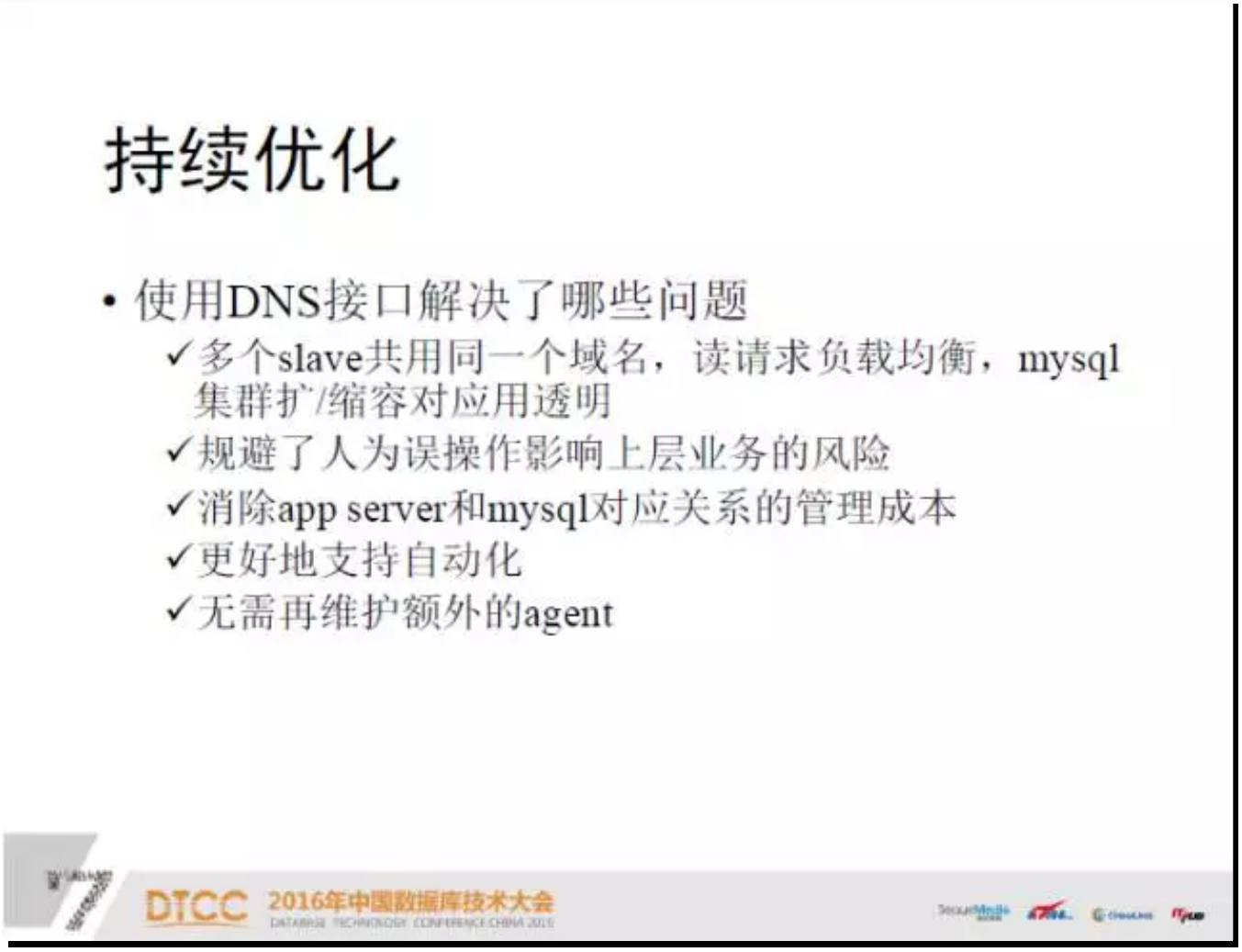
三是把域名当成一条记录。四是可能有一些DNS是内部服务，只需要在内部解析，一些外部的需要到DNS server上解析，以下是改造之后的框架：



register注册到DNS server应用服务器上的MZAgent换成了Dnsmasq，它会向内网Server发送一个解析请求，解析分配给Mysql的内网服务，解析完之后，上层应用再通过IP创建Mysql文件。



DNS cache可能会带来一些问题，比如做一个Mysql切换，24个小时之后才生效是肯定不能接受的，怎么解决呢？可以设置一个比较短的TTL，比如3秒、5秒，强制它解析，二是切换时，主动清除记录，主动purge cache记录。



使用DNS，我们解决了一些问题，首先就是Mysql扩容或者缩容时，应用不需要修改任何配置文件，大家知道DNS记录，我用同一个域名不需要做任何配置更改，当然这个DNS可以自己添加一些比如读的负载均衡，每次解析随机访问一个IP，这实际上是起到了负载均衡的作用。二规避了人工操作。三是管理层的问题，因为所有配置相同和权威的DNS，配置了相同的路由规则，没有一些个性化的东西。第三，无需维护额外的agent。

今天的演讲就到这里，谢谢大家！

关于DTCC

中国数据库技术大会（DTCC）是目前国内数据库与大数据领域最大规模的技术盛宴，于每年春季召开，迄今已成功举办了七届。大会云集了国内外顶尖专家，共同探讨MySQL、NoSQL、Oracle、缓存技术、云端数据库、智能数据平台、大数据安全、数据治理、大数据和开源、大数据创业、大数据深度学习等领域的前瞻性热点话题与技术，吸引IT人士参会5000余名，为数据库人群、大数据从业人员、广大互联网人士及行业相关人士提供了极具价值的交流平台。

阅读 450 2